LEO I and the BR job

Tim Greening-Jackson
tim@greening-jackson.com

October 3, 2012

Chapter 1

Introduction

1.1 About this document

This document is an attempt to document one of the more obscure pieces of early computer history: the use of the *LEO I* computer system to calculate the distances between all mainland railway stations for the British Railways Board.

The document is principally a result of an interview with Roger Coleman held in April 2008. It was he who was responsible — under the late David Caminer — for the BR "job"

The document mostly consists of the following:

- A summary of the information gathered during the interview. I am sure that much of this may be already familiar to *LEO* enthusiasts and computer historians.
- A transcript of the interview with Roger Coleman. As well as covering the BR job in detail the transcript also contains details of other external "jobs" — e.g. calculations of the annual tax tables for H.M. Treasury, as well as background information on the operation etc. of *LEO I*

The document has been updated — to correct a couple of minor typos and also to remove a couple of largely irrelevant sections following John Graham-Cumming's presentation *The Great Railway Caper: Big Data in 1955* to the Strata Conference in London on 2nd October 2012.

1.1.1 Files from the original interview

Files of the original interview are available from the author and will also be sent to the *LEO Computers Group*. The following are available:

- The original WMA file containing the entire, unedited recording (45 MByte). This is also available as a WAV file (978 MByte)
- The interview (inexpertly) "edited down" to remove the author's preamble and various pops, clicks, pauses and other distractions (719 MByte WAV)
- The edited interview then split in to individual MP3 files each containing a different part of the interview.

- Joining the LEO Project
- Architecture
- Printers Cards and Division
- Valves and Mercury Delay Lines
- Inland Revenue Tax Tables
- Background to Railway Job
- Approach to Solving Railway Problem
- Detail of Railway Calculation
- Finishing Off the Railway Calculation
- Execution of the Railway Calculation
- Registers Floating Point and Wrap Up

1.1.2 Acknowledgements

Thanks are due to the following people:

- The LEO Computer Group, particularly Frank Land, for their encouragement
- The contributors to the `uk.railway` Internet discussion group, particularly Ian Batten and David Hansen (whose excellent summary of the railway's *Common Carrier* obligation was particularly helpful).
- Lastly, but by no means least, Roger Coleman for elephantine memory and his patient explanation of an obscure bit of computing history.

Chapter 2

The British Rail Job

2.1 *LEO I*

Author's note: This section is not an attempt to document LEO I *per se, as that has been done much better elsewhere. Any details of the operation of LEO not related to the BR job are included for the information of the general reader, not familiar with* LEO I *and its history.*

In 1947 J. Lyons and Company — owners of teashops and hotels — sent two of its senior managers to the USA to learn about modern American business methods. Impressed by the potential of the new "electronic brains", upon their return to the UK they built their own computer system — the Lyons Electronic Office — or *LEO* in conjunction with Cambridge University, which had constructed the *EDSAC* computer under Maurice Wilkes and had become operational in 1949. *LEO I* — which was to be the world's first business computer was designed around similar principles to *EDSAC* and went live in 1951.

It had 2K × 35-bit words of memory, implemented using mercury delay lines. Input was from punched card, and output could be either to card or to a printer.

Programs were hand assembled (ie. there was no separate assembler program) and the (decimal) op-codes were written on coding sheets. These coding sheets were then keyed as Binary Coded Decimal (BCD) on to punched-tape, which was subsequently converted to pure binary on punched cards by *LEO* itself. Each punched card could hold 16 instructions.

Although some work was done on floating-point representation, most arithmetic was performed using either integer or fixed-point numbers. Special routines also existed to cope with conversion of numbers to the pre-decimalisation pounds, shillings and pence.

LEO was crucial to Lyons: being involved in all manner of operations from stock control to payroll calculation. In addition to these duties, though, it was also hired out to other companies and institutions which didn't yet have computer systems. Such applications of *LEO I* included payroll, meteorological forecasting, calculation of actuarial tables and of tax tables.

These external jobs were done under the aegis of David Caminer's team. Caminer had been the original systems analyst working on the design of LEO and had overall responsibility for software development.

In 1955 British Railways approached Lyons in order to use LEO in order to calculate the distances between each of their 7,000 railway stations.

Note that there is some dispute over the precise number of stations. In his book, Caminer states 7,000, and this figure is also used in the *Lament for LEO* in the *Daily Mail* of 9th January 1965. Roger Coleman's recollection — and it was he who worked on the problem was that only 5,500 stations featured in the final calculations. Whether these were a subset of the 7,000 and on what basis this subset was created is unclear.

To add further to the confusion, in the January 1957 edition of *Lyons Mail*, there is an tantalising reference to the job which mentions:

> LEO is preparing for British Railways a table of distances between railway stations which will involve 18,000,000 calculations.

This figure would actually be consistent with ca. 4,200 stations assuming that for n stations, the number of distances between all stations, d is:

$$d = \frac{n(n-1)}{2} \tag{2.1}$$

Again, whether this is the number of "junctions" as identified by Roger Coleman is unclear.

2.2 The Problem

British Railways' (BR) requirement was for that the shortest distance between each station in the network over existing lines be calculated, no matter how impractical that route would be.

It is highly likely that the driver for this would have been the pre-Beeching modernisation programme of 1955. The modernisation plan was set out in *Modernisation and re-equipment of British Railways* produced in 1954 by the British Transport Commission.

Section 4 of the plan reads as follows:

> Marshalling Yards
>
> 75. British Railways must provide a freight service which gives shorter transit times than at present, with greater reliability and punctuality in deliveries. Speeding up the service and improving punctuality will at the same time reduce operating costs. Reference is made below to the contributions towards realising these objectives that will be made by improved forms of motive power and the adoption of continuous brakes. There are, however, other major factors, including the re-siting and modernisation of marshalling yards which constitute an essential part of the system under which freight is distributed throughout the country. The existing yards for the

> most part were constructed in the time of the individual companies. They reflect a system of routing designed to ensure that the company that accepted the traffic could keep it on its own system for the longest possible distance, and were not specifically designed to facilitate through working. This means that there are often two or more years of old-fashioned design, situated close together, traffic being 'staged' between them.
>
> 76. Rational methods of operating, based on the use of facilities without regard to previous company ownership, require the elimination of many of these old yards and their replacement by a smaller number of modern yards. In such cases there will almost always be direct savings of considerable magnitude, and there will also be indirect saving resulting from the greater fluidity of movement and the reduction in total transit times.
>
> 77. A survey has therefore been made of the scope for resiting and modernising marshalling yards.
>
> ...
>
> 79. The plan provides for the construction or reconstruction of some fifty-five marshalling yards, which would result in the total or partial closure of about 150 existing yards. It is broadly estimated that these proposals would cost about £80 million.
>
> Goods Stations
>
> 80. The introduction of modern methods of handling goods is an important component in the quicker and more punctual freight service which the railways desire to offer to their customers. Many schemes for the reconstruction or improvement of goods stations are also expected to yield substantial economies, principally derived from the more effective use of labour. Economies can also be realised through speeding up the turnaround of wagons.
>
> 81. In planning for the future it will be necessary to close various old depots and concentrate their work in a smaller number of large, modern depots. This will reduce shunting and 'trip' working of trains, improve transits and concentrate sufficient tonnage at stations to justify capital expenditure on mechanical handling appliances.

This wholesale re-organisation of its freight logistics, combined with the planned removal of the Common Carrier requirements on the Railways to carry any cargo at an agreed cost — often unprofitably — in 1957 are the most likely drivers behind BR's requirement.

Analysis was begun on the problem in late 1955, and the results delivered to BR in mid-1956.

2.3 The solution

BR supplied the *LEO I* team with a series of approximately seven railway junction diagrams. These diagrams show each station in a particular area and

the distances between adjacent station on the network, and have connections off-page to other junction diagrams.

Although it would fit comfortably in the memory of a modern desktop computer, the data representing the BR network would be far too large for *LEO I*'s memory, and so Coleman's first task was (manually) to analyse the BR network diagram and break it down in to manageable regions, each small enough to be represented within the memory available on *LEO I*.

The regions were chosen so as to have as few interconnecting lines at their edges as possible. So, for example, Scotland was a single region because there were only three lines coming south to join the rest of the network. These connections are referred to as *boundary junctions* by Coleman. Most nodes within the network are simple: that is they usually just have two adjacent nodes — thus Manchester Oxford Road station would connect only to Manchester London Road (now Manchester Piccadilly) and Deansgate stations, or are terminii like — for example — Paddington. Coleman identified *junctions*: that is nodes with more than two adjacent nodes. It is Coleman's recollection that there were approximately 1,500 of these junctions, divided between six or seven regions.

The approach used was to calculate the following:

1. The shortest distance between each of the junctions in a particular region.
2. The distances between the boundary junctions in all regions
3. The distances between the junctions in a region to all boundary junctions
4. The distances from the junctions in a particular region to all the rest of the junctions
5. The distances from the various station to the junctions
6. The distances from station to station

This step-wise approach meant that not only could *LEO I*'s memory limitations be overcome, but also that the problem could be solved in parts when time on *LEO I* allowed. The most significant single steps in terms of computer time were the calculations for each of the regions in step 1 above.

At the heart of the solution is the calculation of the distances between junctions in a single region. Although it precedes it by three years, the solution adopted appears to be an elegant and simplified version of Dijkstra's algorithm.

Chapter 3

Roger Coleman Interview

This is the transcript of an interview held with Roger Coleman at his home on 22 May 2008.

3.1 Joining the LEO Project

RC I left school in 1950 and when I left school I went to become an actuary in an insurance company and I did some of the of the exams – the first couple of parts. Incidentally I went to an insurance company because at that time I wasn't aware that there were specific actuarial courses at university, which is partly why I never went to university. but then I got a bit . . . I found it quite difficult to study on my own totally, and one day I saw a two-line advert in the *Daily Telegraph* which said something like: "Mathematicians wanted for computer programming" and it literally was three lines, literally three lines in a small ad and I happened to see that and "write to Cadby Hall" whoever it was.

So I did, and it just so happened that in the preceding weeks and months the Institute of Actuaries had approached J. Lyons Company — because it wasn't even *LEO* then — to ask for help in producing some new mortality tables called the A49 tables. But it was based on mortality tables that they had produced around that time. So they wanted to do all sorts of new premiums, issued to the insurance companies these new mortality tables, and that implied quite a bit of tabulation, mathematics and what have you.

I wrote and I had a background in that line, and it coincided with them getting an enquiry from . . . and they had nobody because all the people there were either Lyons' people: Lyons sort of office people that had been involved in the bakeries or tea-blending or whatever it was; or they'd be mathematicians — mostly from Cambridge — who were involved in things for the National Physical Laboratory in Teddington. So they had nobody who basically could speak the language of actuaries — just the language — to the people who were interested.

So I was sort of attractive to them for that reason, so I think that's probably why they eventually they wrote and I went and there it was,

and so I joined them. I joined them in — I think it must have been in December 1954 at which time I was 22. So I joined them and then I learnt to program.

There was only one computer: that was *LEO I*; and I think — I was trying to work it out — but I think I was probably the eighth programmer when I joined them. I was also the youngest, but I think I was the eighth. Essentially there were no programming courses so I was sort of taught an hour at a time by the guys that were already there, but there was nothing to read.

3.2 Architecture

RC The thing with *LEO I* was that it did have a proper order code and so you did write programs on paper — you didn't have to wire it or anything like that — your wrote it on paper. The programs were punched on to paper-tape, the paper-tape was converted and that was keyed in binary-decimal and the programs in that form were fed in and converted to punched cards. So from then on apart from the initial conversion from writing to paper-tape that was the only time paper tape was involved. So after that, the programs were all on punched-card; and the punched-cards were binary cards.

So on a normal 80-column card — from memory — you could get 16 instructions, because they were half the card. The instructions was 16-bits split in to three parts: an action like clear or transfer; the second part was the address to which it applied — where the thing was that you were going to add or subtract.

TGJ This was a memory-address presumably?

RC It was a memory address and the memory was 2,048 16-bit words. Now that had to hold all the "operating system", the program and the data you were working on. Now the "operating system" which we wrote — in fact I wrote it for *LEO I* — was two binary punched-cards, and you fed these cards in to the card-feed and pressed the `START` button on the console. All that did was to read the first card and go to the second instruction on the card; and the second instruction was "read the next card" which was the other half of the "operating system". So you then had all the "operating system" inside the machine, and it then went on from that to read in the program — which was stored behind the operating system on punched cards and had various parameters to tell the machine where to store things etc. Having stored the program it went to the start of the program, and the first instruction in any program was `STOP`, because otherwise it would have launched straight in; so instead it got to `STOP` which meant it was ready to go but stopped — so your program was in the machine but stopped — while you decided whether you wanted to put some data in to the card-feed etc.

The general input was punched-card, and certainly from a programming perspective they were binary cards with 16 instructions per card. The paper tape was fed in and converted to punched card, but you got quite

used to filling up the holes: you didn't punch the cards again, you just fiddled the [chad] in with your nail. It was far quicker, and that's how you altered instructions.

TGJ A couple of questions arising from what's just been said. Firstly you wrote the instructions down on a pad?

RC From the flow-chart. You need a flow-chart first.

TGJ These were then punched on to tape by an operator presumably?

RC Yes.

TGJ Those instructions presumably were like assembler mnemonics [such as] `ADD` or `MOVE`?

RC `ADD` was 14. Actually "clear the accumulator and add something in to it."

TGJ So you actually wrote *14* rather than *ADD*?

RC I did. 14 and a numerical address, but it was an address referring to the third [part of the instruction] which was a section of the store you had set up when your wrote the program.

So in other words it might be that all your constants were in section 4, and somewhere you wrote where section 4 started. So in fact you never had to worry — when you were writing the thing where it — would pitch-up in physically in the computer. You sorted all that out afterwards and eventually would say something like "I want section 4 to start in 609" of the 2,048 words.

TGJ So within a particular instruction you would just refer to Section 4 and an offset in to that section?

RC It would be "item 4 in section 5" or whatever.

TGJ So you never actually referred to an absolute [address]?

RC No.

TGJ That would actually make for a lot more efficient addressing.

RC It was a damned sight easier, because you didn't have to keep going back. It was actually a good system — quite clever really. Now I didn't invent that — that was all part of the design . . . there before me.

It made writing programs easier as you didn't have to think while you were writing where anything was going. So anyway that was broadly the input. Most of the programs were stored on binary punched cards, and that's how you stored them in filing cabinets. The output was punched-cards and you could punch in binary.

3.3 The Line Printer, Punched Cards and Division

RC There was a line printer, I can't remember what model it was but the bars came up and they went *CLONK*, and the bars went down and went *CLONK*. And the *CLONK*s were probably about a second apart. It was quite quick relatively.

TGJ 60 lines a minute.

RC It was maybe quicker than a second... It's a long time ago remembering it; but you didn't get the feeling it was taking forever — put it that way. You could produce a sheet of continuous stationery quite quickly.

TGJ You had continuous stationery?

RC Oh, yes.

TGJ Was that specially produced?

RC Probably. I'm not sure because there were Hollerith tabulators in those days, and [they used] sprocket-fed continuous stationery. I suspect as they did payrolls on Hollerith punched card systems and they still printed out payslips on continuous stationery.

TGJ Presumably, say you are doing something like payroll and you have a limited amount of physical memory - you've got 2,048 words — so presumably if you are doing payroll or whatever you would have a stack of punched cards [one] for each employee

RC You had brought forward cards for each employee and at the end of the processing of that guy you print out the updated card which you bring round next time and feed in

TGJ But for jobs where you've got significantly more input data than you could possibly store at one time in your memory you could deal with that by having a stack of cards.

RC You did You had a stack of cards and you read them in one at a time if you were dealing the next blokes pay. You deal with him and then he gets printed out... I can't remember so much about payroll. I don't think they printed them at the same time, as they had to do all the reconciliations and stuff before they could actually risk printing anything. But that was never my subject.

TGJ Payroll was an arbitrary example

RC I was never involved with Mountains of printing.

TGJ I started programming in the 1980s when you had dumb terminals attached to a VAX system

RC Oh vastly beyond ...

TGJ ... now is a museum piece in its own right. Something the size of a chest of drawers which sat in an air-conditioned room. Now obviously everybody has got a workstation under their desk.

RC With more capacity than I ever had. Ever.

TGJ One of the things that strikes me that the voice recorder on the table has actually got several thousand times the storage capacity than *LEO* had and more processing power.

But it [also] strikes me that there had to be a lot more ingenuity in the way that problems were addressed [in *LEO I*'s day].

RC That's right. Because You had so little space to do anything on. So nowadays clearly you don't even think — and if it takes another gigabyte then what the hell. It's cheaper than spending lots of time thinking of clever ways around it. It's just cheaper so in a way you could say "Why the hell think? Just throw capacity at it." But it wasn't like that then.

TGJ The other thing that also occurs to me about the time was that there was still so much to be discovered and so much to do. So things that we take for granted... if I were writing software to do this sort of calculation then I have already got libraries of trigonometric functions for example at my disposal which wasn't necessarily the case and all sorts of stuff that I would take for granted which you would have to write from scratch.

RC That's absolutely right. The standard division routine on *LEO I* was a separate subroutine. There wasn't a divide instruction. You had to call in a subroutine to do that. You could do multiplication, but not division.

The other thing that you have got to remember is that this was in the days before decimalisation. So you actually had to reconvert binary numbers that were money in to pounds, shillings and pence. You probably hadn't thought of that. So there was a de-convert instruction in to decimal and a similar thing in to sterling. So that was another thing that you had to bear in mind.

The whole atmosphere was much more... you knew everybody. There was an awful lot not written down. There weren't standards — well there were some standards — but it was much more like a research establishment. There was far more ingenuity, and there had to be.

TGJ Something else I'd like to get your take on obviously you've got 2,000 words of memory and effectively a processor...

3.4 Valves and Mercury Delay Lines

TGJ ... I'm presuming in the early 1950s it would still be valves rather than transistors.

RC It was valves, and the storage was mercury delay lines

TGJ That's something I wanted to ask you about. I remember reading a book [...] over 20 years ago when I was at university about early computers

in Manchester, and that mentioned mercury delay lines where you had a quartz window on top of a steel tube [filled with mercury].

RC In *LEO I* they were horizontal [...] Now I'm no engineer, but the whole idea was that you had these 0s and 1s zooming round the loop at the speed of light. However, you had to slow them down, otherwise you couldn't get at them. And you slowed them down by passing them through these mercury tubes, and the 0s and 1s rippled down through the mercury and that slowed it down enough so that you could actually extract stuff from them. Of course they had to be at a certain temperature and so they were all under the floor. They were probably about [four or five feet].

TGJ But that was actually how stuff was stored?

RC The stuff that you are operating on — the 2048 words that you could get at and instruct an get it out — they were all stored in these circuits, going round and round and round; and they were controlled by these valve things. Of course the timing was critical — because if it wasn't absolutely spot on you got the wrong bit out. Which is why there was so much temperature control. That was physically how it was done, now I never was an electronic engineer, so I can't tell you an awful lot about that. But that's how it was done.

TGJ It's very seldom I'm lost for words. I was aware of mercury delay lines. In computer circuitry sometimes you just need to delay a signal for some reason. I wasn't aware it was a means of storage.

TGJ So you have a machine with so many thousand valves,

RC Racks of these things...

TGJ So I imagine that the heat must have been ferocious.

RC Well it must have been, but I can't remember. I don't remember it feeling particularly hot, so it must probably was air-conditioned.

TGJ The other thing is that a valve has a finite life-time...

RC What I can tell you about that is that every day the engineers had the machine for an hour. They tested under margins and anything that was likely to break would break when they increased — I don't understand electronic engineering and never did but as I understand it they — so to speak — put the load up a bit, and if anything was a bit on the blink it would go then, and then they'd change it; if it didn't go then. Then it would go back to normal what I would call "operating pressure" and then it would be alright for the rest of the day. So there were breakdowns, but given that, it was reasonably reliable.

TGJ Thinking about it I suppose it had to be given that it would be in use every day by Lyons tea-shops and then at night for the government or the railways or....

RC For all sorts of stuff. Of course it wasn't just the tea shops because they did all the tea blending. A tea-blending exercise. That was probably twice a week and they did the payrolls for all the Cadby Hall people. There was quite a reasonable work load. They certainly couldn't be out for days and days — it would have caused havoc and I don't remember it ever being like that. I was there from 1954 until end 1957. I was much more involved in the non-Lyons [jobs].

3.5 The BR Job

3.5.1 Background

RC There are two things about the railway job that you have to remember. The main one is that I think there were 5,000 stations. You put a number [forward].

TGJ I've heard two different numbers. I've heard 5,000 which I think was on the radio programme; and I've seen a reference to 7,000 in the book on *LEO*.

RC I think it was nearer 5,000. If you'd have asked me and I hadn't seen these numbers I would have said 5,000 to 5,500. You've written:

> I understand that there were 7,000 or so BR stations, so by my calculation this would mean working out 24,500,000 separate distances

How do you get that from 7,000?

TGJ Right. Imagine you've got n stations. So from the first station you have n different distances to work out, from the next you have $(n-1)$. It works out for n stations as

$$d = \frac{n \cdot (n-1)}{2} \tag{3.1}$$

[Where d is the total number of distances] Trust me it's right ... if you want to work out the distance from every station to every other station

RC Which we did.

TGJ I want to come on to how you broke the job down... how you solved the problem.

RC The other thing that you've got to remember is that — I think this all happened after Beeching had closed down or was closing down an awful lot of the network. But the main thing that was significant to how you do it was that the brief was that we had to provide the shortest distance over existing railway lines between nay pair of stations.

TGJ That's an interesting restriction

RC What you didn't have to do was to say how you [routed it]. That's what it was. They knew that you wouldn't take it that way. They didn't give a stuff how it was actually taken, because it would probably go via marshaling yards and all over the country. That was irrelevant — the charging system was the shortest distance over existing railway lines. That was the phrase. So nobody was bothered about what the route was.

TGJ But the other important thing is that it's actually over... I naively assumed that it would be... You've heard of a great-circle calculation

RC Yes. But it wasn't. No. Nothing to do with that. Over existing railway lines.

TGJ It then becomes... are you familiar with the travelling salesman problem?

RC Not as such, like that.

TGJ It refers to a class of problems in computing where you need to visit x number of places and you work out the shortest route to travel. You have the same thing now in things like Multimap or GPS satellite navigation. It will work out the shortest distance on known road. But again, in the 1950s that would be unknown.

RC It was over existing railway lines, that was the brief.

RC So the first thing I was given by a chap in British Rail called Hanky. Mister Hanky.

RC Did you ever meet Caminer in *LEO*?

TGJ No.

RC You've come across the name David Caminer in *LEO* terms.

TGJ Yes.

RC Well anyway, he was sort of the big boss of programming and he always called him Old Snodgrass. "Have you spoken to Old Snodgrass." Anyway, his name was Hanky, and he gave us the map of British Rail. Now the map of BR was not a map of England or Scotland or whatever with the lines on, it was much more like underground maps, and there were sheets about $[1.5M^2]$ but they were in diagrammatic form, a but like a London Underground map.

TGJ So the distance between the stations on the map bears no relationship to the geographical distance

RC Not only that, there were about six or seven of these maps covering England Scotland and Wales — the mainland — and there were overlaps. So I took them home and I spent about three days at home finding out where the overlaps were. You couldn't literally overlay them because on one map there might be a line [at one angle] and on the next map it might be [at another angle]. So I took these maps home and sorted out what we'd got.

3.5.2 Approach to Solving the Problem

RC Broadly the approach was that we split the country in to regions because clearly we couldn't get all the stations in the computer at one go. We split it in to stations and junctions. Now the definition of a junction is where you have an alternative route. In other words if you have a line going along a valley and there was a spur going up and there would be a junction so to speak at the spur, that wouldn't count as a junction [in our definition] because it didn't give you an alternative route to the rest of the country. So a junction in this definition was a place or a station from where you had a choice to other parts of the network. So that's the definition of a junction and there were 1,500 of them out of that 5,000. That's all.

Because the argument is that the stations, if you are on a normal station you've either got to go [one way] or [the other way] So that's not so bad. So the key is the junctions. We couldn't get 1,500 of these things in the machine either. So what we did — and this was my first month — we split the junctions in to regions.

So you had a region of junctions, and another region, and another region. What you had to do was to split these regions in such a way that you had the minimum number of *boundary junctions*. And a boundary junction was a junction where you went from one region in to another region. So for instance what you found was that if you take Scotland, I made Scotland a region. Now the big advantage of that was that — I don't know hat it's like now — but there are only three ways by rail from Scotland in to England. There are three lines: the East, the West and there is one down the middle. So in other words by making Scotland a region, I only had three boundary junctions. Right. So, I broke down these 1,500 junctions in to — I can't remember — it was sort of six or seven regions. Each boundary junction will be on a boundary of two regions. Then the data they gave us was basically from every station [to] its next-door in miles. That was the data they gave us. I can't remember if it was on the maps. I don't think it was on the maps — I don't think I extracted all these millions of numbers. I think they gave us that on sheets. But that was the data. So then what we do. So I took each region and I'll show you now how it works...

3.5.3 Detail of Calculation

RC What I've done is, I did this just out of interest after you called, because I wanted to see if I could reproduce [the solution]. So what I've done, this is an arbitrary region and I haven't put the stations on, So I've got the junctions which I have called `A` to `K`, and the thing is you'll find that every one of them has got a choice of ways in to the rest of the thing. In other words from `D` it's to `A`, to `B` to `F`. Or from `G` it's to four of them in that case.

But they've always got [at least] three because otherwise if it had less than three it wouldn't be a junction. Now there's the data. So from junction `A` it goes from `B, D, E`. So what you do you set up in the machine `A, B`

...K, and it's worth spending a few minutes because this is the key to the whole damned thing.

TGJ And each of those is a junction?

RC That's a junction. What you do is you set all the distances by going to pitch-up with the minimum distance from one of them. Lets assume that this calculation is to go from *G*. So the first things you do is you set all of these to be 100 miles. 100 miles because I know that there's nothing going to be 100 miles within those sorts of numbers.

Now I'm going to calculate the distance from G. I'm going from G, so I'm going to set that at 0. Now I go to each of these. Now where's G connected to? G's connected to F. It's 1 mile. Is $1 < 100$. Yes. So you do two things. You set that at 1 and you open [F]. A tick means that's now an open junction. The next one. So that's that. H that's 1 and open it. You will see that this is a programmable operation. J is 3 and open it. K [is] 3 and open it.

Now we've finished with G so we close G ... and I move down the list to the next open junction: H and I've got 1. H goes to C which is 4. $4+1=5$. Is $5 < 100$.

TGJ So the most efficient route we've found so far is 5 miles.

RC I'm not even looking at that. So that's H. I've done that one. To G is 1. $1 + 1 = 2$. Leave it. Of course that's itself and there and back.

To K is 1. Were working on H, 1. $1 + 1 = 2$. $2 < 3$ so we change that to 2 but this is already open so you don't do that. Now we've done all the stuff from H so we've finished with H for now and we close it. We move on to the next open junction which is J. $3 + 3 = 6$. $6 < 100$ and open it. F is 2 so to F it's $3 + 2 = 5$. $5 \not< 1$ so you leave it. $3 + 3 = 6$ to G. No, leave it. $3 + 2 = 5$ to K. No so leave it. So we've now done J so we close that. [The] next open one is K. K to J $2 + 2 = 4$ $4 > 3$. $2 + 3 = 5$, $2 + 1 = 3$. So leave that. Nothing more.

Start at the beginning again. [The] first open one is C. $5 + 2 = 7$. $7 < 100$. $5 + 5 = 10$. $10 > 1$, leave it. $5 + 4 = 9$. No so leave that. Next open one F. To D is 1. $1 + 1 = 2$ and open it. $1 + 5 = 6$. No [$5 < 6$]. $1 + 1 = 2$. No. $1 + 2 = 3$ that's the same so you leave it. $1 + 3 = 4$. $4 < 6$ but it's open already so you don't have to reopen it. $1 + 4 = 5$ now we've just been dealing with F haven't we. So we keep on going. I $4 + 2 = 6$, leave it. $4 + 3 = 7$, leave it. $4 + 3 = 7$, leave it. Back to the beginning.

TGJ Of course as you get further through the computation you get fewer and fewer to replace you're always approaching the solution.

RC The next one we open is B. $7 + 2 = 9$. $7 + 2 = 9$ leave it. $7 + 2 = 9$, leave it. Next open one D. $2 + 3 = 5$. $5 < 9$. But its already open so we don't have to close it again. It's easier with a machine actually. We're on D $2 + 2 = 4$. $4 < 7$ and re-open it. $2 + 1 = 3$, that's ok leave it. Finished with that. $5 + 7$, leave it. $5 + 4 = 9$, leave it. $5 + 2 = 7$, leave it. $1 + 1 = 2$, [the] same. $1 + 5 = 6$, leave it. $1 + 1 = 2$, leave it. $1 + 2 = 3$, [the] same, so leave it. $1 + 3 = 4$, the same, so leave it $1 + 4 = 5$, [the] same so leave it.

We're nearly there. Back to the beginning. Right `A` $5 + 2 = 7$, leave it. $5 + 3 = 8$, leave it. $5 + 7 = 12$, leave it.

TGJ We're done aren't we then?

RC No we've still got this one so we open that again. $4 + 2 = 6$, leave it. $4 + 2 = 6$, leave it. $4 + 2 = 6$, leave it. All closed. And those are the shortest distances from `G`.

Now I haven't been looking at it, but I think that's probably right. 1 to `H` 1 to `F`. That's the same whether you go that way or that way, but it's still 3 to `J G`.

TGJ The test case will be `A`, `B` and `C`

RC It's $1 + 2 = 5$ `G` to `C`. $1 + 4$, $1 + 5$. So that's right. You can see that's programmable.

RC The interesting thing is that if you look at computer networking that's basically the way that least-cost-routing works in networks. Imagine you have a very complex network. Each hop of the network you allocate a cost to. That's a precursor to what would become... effectively the railway is a network anyway. Least-cost-routing in the real world.

RC That was the thing I programmed first. So just to finish off. I did each region in turn. Then I did the same calculation just with the boundary junctions over all regions, because I'd got ... I cant remember how I did it but basically it was the same principle but I then had the shortest distances over all ... just the boundary junctions, were you went from one region to another. Which were probably about 100, from memory.

3.5.4 Finishing off the Calculation

RC But then it was no longer doing it this way. I did it from each junction within a region to all the boundary junctions. Then I did it from each junction to all the other junctions. I then got from each junction to the all the boundary junctions. Then I can go the other way round. So I got all the regions to all the regions, but it wasn't this sort of calculation any more. Then after that it was every station to all the junctions, because that was just a choice of two, because you just went well I've got the shortest distance. There's a station and you've got two places. I know the shortest distance from... I've got all the junctions to junctions, the shortest distances, so all I've got to do, is there's one end bit on to each one and what is it. Then equally with stations, the station to station. That's equally quick. There's more of it but it's still only one to one.

TGJ Out of curiosity, how did you manage with London... I presume that it would be a region in it's own right?

RC No I'm not sure that it was. I don't remember is the quick answer; but as far as I'm aware it worked exactly the same. If you take for instance Kings X, that was probably a station not a junction at all because certainly in those days, you had to go out of Kings X ...

TGJ Yes of course. It's a terminus, isn't it!

RC ... to the first point where you did have a choice. Now with something like Victoria it's very quick when you leave Victoria to get to the first junction — it could well be Clapham or something. But it wouldn't be Victoria itself nor would it be Waterloo, nor Euston nor any of those. In this context they are stations

TGJ So if I come out of Euston I've got to go up to Camden before I decide whether I'm going up the fast line or the slow line.

RC Absolutely, and of course the other thing was because it was done a region at a time and then it was broken down like this you didn't have to do it all in one go. One night you'd do a region with whatever it was. All this output like where that would go on say a punched card from `G` and that would be from `G` to that network. So that would be the data from `G` and that would be stored on punched cards. I think there were 1,500 junctions and I think there were about 100 boundary junctions.

TGJ So presumably you would do one or two junctions a night. Well it depends which stage of this you were at. You had to do each of the regions first. You had to do that. And one of the things you would get from there, in this thing `C` to `J`. Now that's a boundary junction and that's a boundary junction. And you wouldn't worry about `G`, `D` and `F`. That's why the key was to have your regions [with] as few boundary junctions as you possibly could, because every extra boundary junction gave you a way in to the rest of the world.

TGJ Actually thinking about the little bit I know about the UK railway network, I suppose it would naturally lend itself to being split in to regions anyway.

RC Scotland is a case in point

TGJ You've got Scotland where you've got Glasgow, Edinburgh, Stirling etc. have their own bits of network but then nothing very much.

RC Well there were three. It was quite complicated around Glasgow and Edinburgh. Given you didn't have to do it, you wanted to keep it to three to get to the rest of the world. Obviously now I can't remember — I remember the three in from Scotland — because that's an obvious one; and I suspect coming up from the West Country. There aren't so many ways out of Wales I suspect. I didn't particularly look at the geography in terms of Wales or Southern England, but it worked out like that and the bottom line was that you couldn't have more than whatever it was, otherwise you couldn't fit them on the machine for doing this sort of thing.

TGJ Presumably when you were planning the solution — I don't mean the algorithm, I actually mean executing it you then just broke the network up in to bite-sized chunks and did so many

RC As I said you did each region at a time. You have to work your way through all the regions first, separately.

TGJ But you couldn't do an entire region in a single run though presumably?

RC You could do this for Scotland in a single run. If I hadn't been able to I would have had to have broken it down in to two different regions.

TGJ Sorry yes. You're perfectly correct.

3.5.5 Finishing off the Railway Calculation

TGJ In terms of execution time, how long did it take to execute?

RC I can't be sure at this range, but what I can say is that when we were doing a region like — say — Scotland my instinct says it absolutely wasn't more than probably an hour. Actually when you think about it, that's a very straightforward bit of loop.

TGJ It's just adding and subtracting...

RC Testing against the previous number; opening and shutting. It was doing — I can't remember how many instructions per second it was — but it really was quite ...

It wasn't searching for anything, it wasn't looking up all sorts of things, it was just plodding on, and I think that the regions probably... You were talking hours not days.

TGJ So you could set it running overnight and the region would have been done by morning.

RC It wouldn't have ever been that long. You wouldn't have ever done that. It was shorter than that.

TGJ So you could set it running have a cup of tea and...

RC Well certainly it's a late evening, but it's a late evening rather than an all-nighter.

Once you're going from station to station say to all the other stations you just do ... if you've got an hour, well I'll do the next dozen stations. It doesn't matter any more. Once you've got the framework, first of all of individual regions, you have to do those on their own. You have to do the boundary junctions to all the other boundary junctions as one. But after that you can do it much more piecemeal.

TGJ In terms of the amount of analysis — we're getting back to ingenuity — it's not simple at all.

RC The biggest thing was we weren't given this solution we were just given the bloody maps and said what can you do? As far as we knew there were no precedents to look at. It was basically us thinking.

TGJ This was done 1952 to 1955?

RC No this was done between... I got the maps, spent my days at home in the Autumn, the last quarter of 55 and the whole thing was done some time around the middle of 56. I can't be certain of that

TGJ So nine months between the problems coming in the door and being signed off. I suppose a goodly proportion of that would be the analysis as well

RC Well, quite a lot. There were only two of us doing it. It wasn't vastly expensive

TGJ I am genuinely impressed at the ingenuity of the solution

RC We thought it was quite clever. Also I'm impressed that I can reproduce that after 55 years

TGJ When you look at it, it is the only logical way to do it. With a lot of these solutions when you see it you think "oh that's really obvious" but actually thinking of it without someone showing you is hellishly difficult.

RC That's right. I guess we had ideas and fiddled.

TGJ I'm hoping to get this from Kew, but as I understand it the driver behind it — this is according to the book on LEO — was some sort of statute or something.

RC It was something. It had to be charged on this, as I said, I suspect although I don't know it was tied up with the Beeching closure of all these lines. It was about that time. I suspect there was a connection in some way, but I can't tell you what it was.

They were good time, you know. They were good times for a young lad more or less just out of school. Very interesting.

TGJ It's a completely different problem to the one I expected.

RC Well I know. When I read [your E-mail] and you mentioned great circle [calculations] I thought "Great circle!? What!?" I can see how you got there but I thought "What?"

TGJ It's a very interesting problem.

RC Well I think it was. It always struck me as an interesting problem, because the great thing was it is so easy to set out what the problem is. You can set it out in about thee sentences and everyone understands what the problem is. I think that's one of its attractions

TGJ How many regions were there?

RC I think it was probably around seven. I can't be certain.

TGJ So there would be about 200 junctions in each region

RC I'm pretty sure there were about 100 boundary junctions — I think — but again I can't be certain of that. I'm pretty sure about the 1,500 junctions (by my definition) out of those 5,000.

Again the algorithm you've got there, [...] for each point in the network it considers all the other points, the total number of calculations will be proportional to the number of junctions — it will be order N^2 — about 40,000 sets of calculations per region.

Yes. But it's an iterative process. You come back to a place. You're not looking at one and then doing calculations. You're there. As you see we sort of reopen things. There is a sort of iteration. I think you'll find I deliberately picked my source as one in the middle, because f you took probably this one you find you don't have to [...] re-open anything. That might just be of use because it's:

1. Regions in turn
2. Boundary junctions to boundary junctions over all regions
3. Junctions in a region to all Boundary Junctions and then
4. Junctions in a region to all the rest of the junctions then you go to
5. Stations to junctions and
6. Stations to stations

That was the sort of six parts — six sets of programs in a way

TGJ Then effectively it's just the output of 5 and 6 that British Rail were interested in.

RC Well in fact [just] 6.

TGJ It does make a lot of sense it being connected with the Beeching programme, because it would show the quickest or shortest or cheapest path from A to B

RC I suspect so. It was simply that nobody could argue... if they are being charged on the shortest distance then nobody gives a stuff which way it physically goes; and nor do you have to worry about which way it goes. I suspect before — and this is my suspicion rather than anything else — that they sort of charged it from one marshaling yard to the other and then from there and somebody quite reasonably said "Sod this for a game of soldiers: this is absolutely pointless and we've closed so many lines we need to do it all again anyway" I suspect it was tied up with that.

TGJ I'd also heard that is was to do with the introduction of a distance based fare structure.

RC If that was true, that's what I've said: the rules said you have to charge for the shortest distance over existing railway lines. Now I as far as I know I should have put in brackets "at the time of calculation", because it was never updated and nobody was interested that I now if in dating it; and I think I would have heard even if I wasn't there and anyway once LEO I had gone they would have to have programmed the whole bloody thing again, because it wouldn't have run on anything else.

3.5.6 Registers, Floating Point and Wrap Up

RC That's the background. We never had FORTRAN. We'd never heard of FORTRAN then.

TGJ It was [still] in the process of being invented in 1955 by a guy called John Backus at IBM, who died a couple years ago. He was one of the big names in computer sciences as he worked out a lot of the rules pf computer languages. Yes. I think it's remarkable writing it in effectively machine code is amazing.

RC It wasn't quite machine code, because we had these relative addresses. It was in terms of ... there was one accumulator and that's where you put things in to, operated on them or subtracted or multiplied, and then somewhere else...

TGJ Again you must excuse my ignorance but in terms of what I would regard as registers on a processor where I have usually a number of them where I can keep different — you know — 32 bit or 64 bit [quantities]...

RC No.

TGJ There was just the single accumulator

RC Yes. The only other thing there was, was if you were multiplying there was a separate place to put one of the two [operands] then you specified the instruction was multiply and the address was the number you were going to multiply. That's where you put the two parts of the multiplication. The first part you had to put in a register first or whatever they call it now. Then the instruction was multiply which was either `21` if it was negative multiplication or `31` if it was [positive]. I even remember that! And then the answer from the multiplication went in the accumulator.

Out of curiosity, the codes to which you've been referring the `21`s etc. were they octal or decimal.

They were binary up to 31. There were 5 binary digits which gave you up to 31. That was the operation. So you wrote down `21` on your sheet and that was keyed on paper tape as `2` in binary and `1` in binary, which was converted in to `21` in binary when you fed it in

Was that why you had the two-stage paper-tape to punched-card?

Yes. That happened at that stage. So then you pitched up with an instruction on the card — a binary programming card. There were 16 instructions [per card] because there were 8 rows and two spare rows and then they were halved. That ended up as an instruction on one of the programming cards.

TGJ How did you handle non-integer numbers? Did you have floating point or did you use fixed point?

RC There was floating point. I never really got involved with floating point. But they used to use floating point. The mathematicians that were doing things like that, where they had a number but they also knew where the decimal [point was]. The problem that you did come up against was accuracy in the sense of if you were rounding numbers to the penny in a lot of things like tax tables or you were multiplying that sort of thing you had to be careful. If it mattered — like it did with some of these actuarial tables, the last digit [was significant]. You had to make sure that you'd

looked after, you'd considered the whole business of rounding errors, and what the rounding error could be and how if you had to go the odd extra digit in order to get the rounding right. That was a bit of an art in itself and you got used to that. That was just what you did.

I suspect that there's no documentation at all — or at least that's what I suspect; or there's no documentation that's actually relevant [to] what we've been talking about. I'm sure there will be something in the archives that says that British Rail paid a couple of thousand quid for it to be done or whatever.

TGJ But in terms of the correspondence from Mr. Hanky...

RC Absolutely. I doubt if there's any of that. Besides which, most of it wasn't done by letter — it was all done by talking. We never exchanged letters, so I'd be most surprised if you found any correspondence other than a bit of contractual type stuff, and you might find that. I'd be interested in that too.

TGJ Yes. I will certainly ask the question of Kew, and see what they can come up with but don't hold out any particular expectation

RC You know, don't you that the job would be always referred to as 'O51'

TGJ Zero or the letter O?

RC No. O for Outside. Not zero.

TGJ Referred to by British Rail?

RC That's what we called it — it was our internal job O51.

TGJ O for outside

RC Outside, Absolutely. O for outside, L for Lyons, E for...I can't remember what E was for. But anyway, I'm sure that any letters would somewhere on them would certainly from us would have had O51 on them. I'm sure that if they'd put anything they'd have the same reference number. At least it's worth...if you see anything with that reference on it's probably to do with it. At least it's worth knowing.

TGJ Well, thank you very much

RC It's a pleasure.

Appendix A

Artefacts from the interview

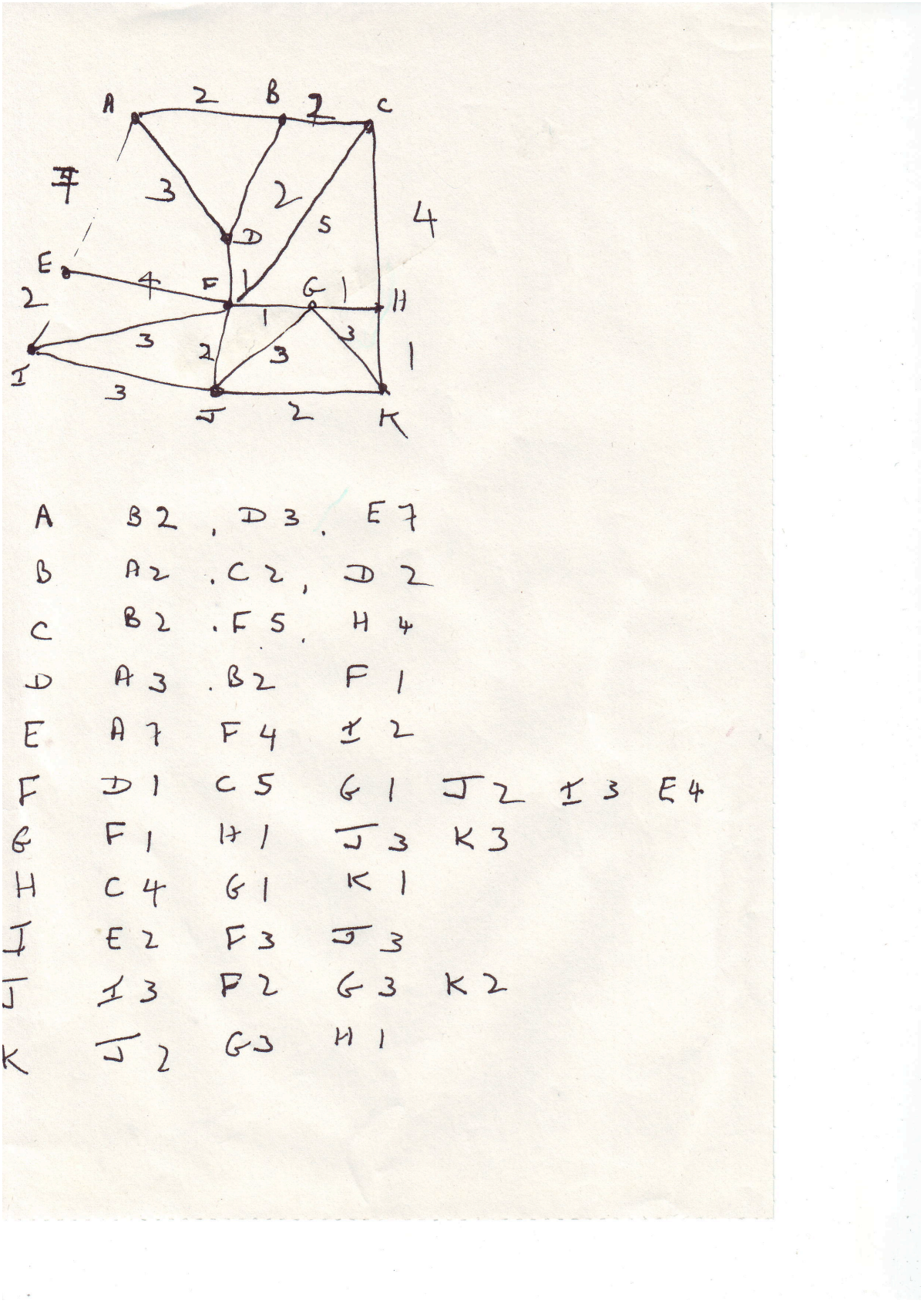

Figure A.1: RC's sketch of arbitrary network

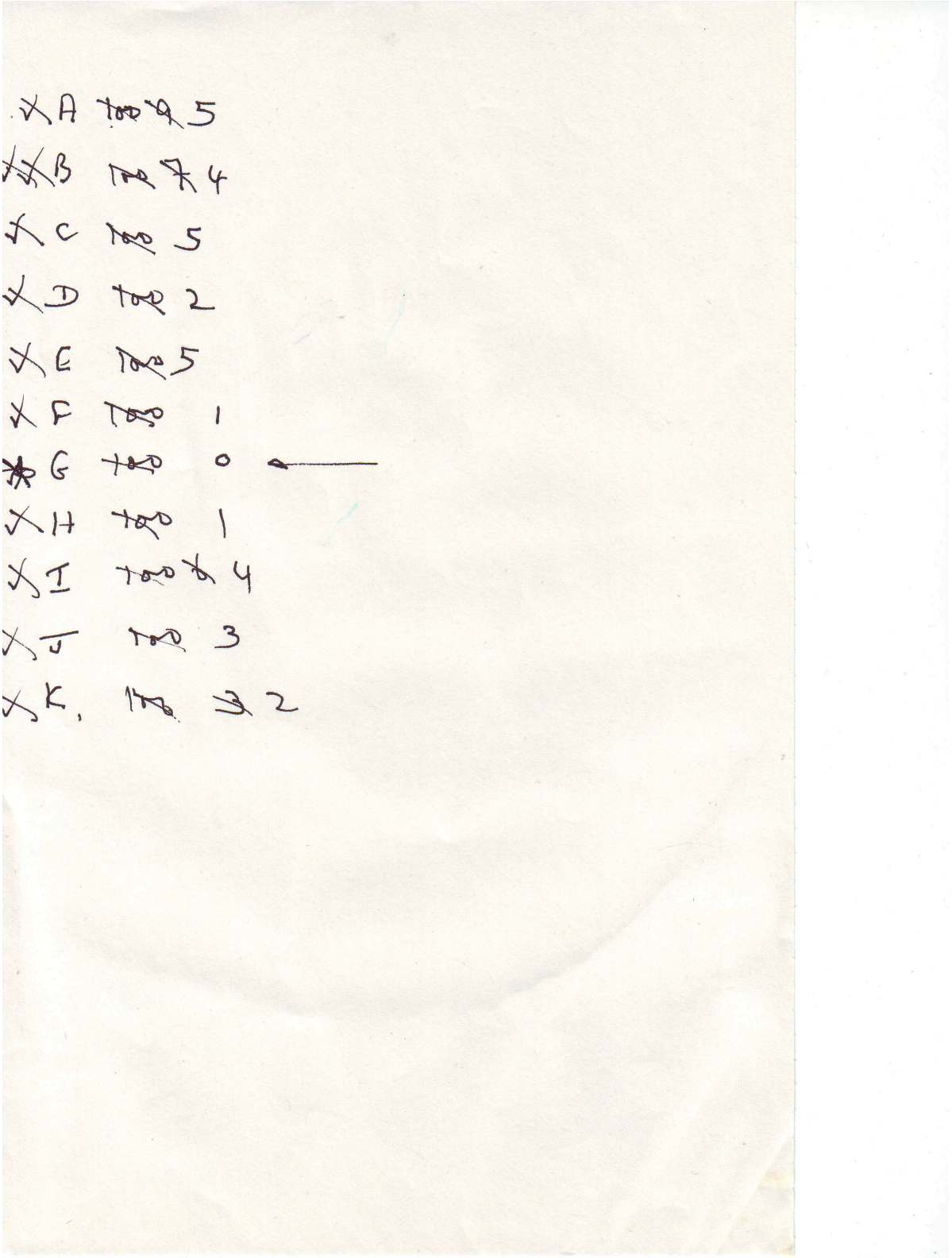

Figure A.2: RC's calculations for arbitrary network

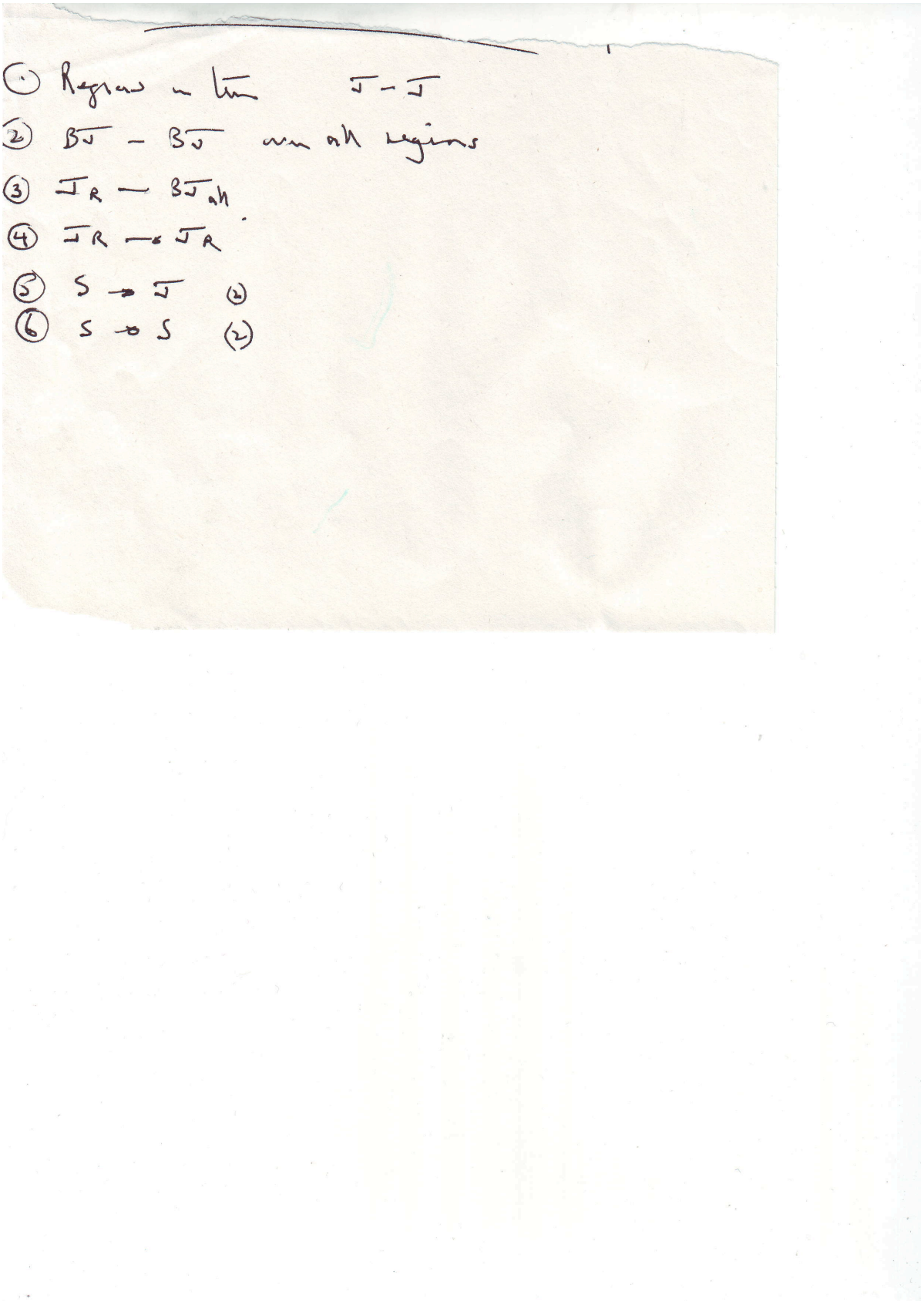

Figure A.3: RC's notes on order of evaluation