Algorithmic Problem Solving: Introduction

Fall 2014 and Spring 2015

Given by the Natural and Formal Languages group

August 20, 2014

Welcome and Overview

- The scheduled activities for this course is a *round-table meeting* every Friday at 13:15.
  - The current meeting is the only classical lecture given
  - All following meetings rely on *student participation*. We will give short lectures, explain some techniques, and give some examples, but student discussion is expected.
  - Discussing students own chosen problems is the core goal
- The mix of teacher-led and student-led discussion will be somewhat flexible: expect an average 50/50 split
- The majority of the coursework is independent problem investigation

Reading Pace

This course can be read at different paces:

1. At 25%: take it the entire fall term.
2. At 12.5%: take it over both fall and spring 2015 terms.
3. Inofficially 50% (half a term) is more or less possible by doing most of the work in during a short period. This requires attending some extra Friday meetings outside the 50% slot.

Completing the Course

Each student individually need to:

1. Attend and *actively participate in at least 12 full round-table meeting sessions* (this lecture counts). Each term has more than enough meetings (14–16).
2. Choose a problem to work on, trying to solve/analyze it from an algorithmic perspective, and:
   - Discuss it at meetings, getting help and suggestions.
   - Give at least *two longer presentations* during meetings. These should be well prepared, take 20-40 minutes, and discuss your chosen problem.
   - Write a report detailing your problem, work and results.

The course plan suggests that meeting attendance is 20% of the course, the presentations is 30% and the final report 50%. In practice working on the problem is spread across all three.

The possible grades are $U, 3, 4,$ or 5.

Choosing a Problem

Choosing a good problem is tricky.

- Problems tend to be too easy or too hard
  - Start by trying trivial solutions. If it is too easy: write it up anyway and start another!
  - If it appears too hard, simplify! Pick an aspect that seems solvable and iterate. Many problems have "too many moving parts": fix or remove some parts.
- Discussing in the group often quickly reveals whether something is easy, hard and/or interesting.
- Check the home page for suggestions, but mostly, ask and discuss your interests!

A First Problem: Paying Debts

Going to the mountains Martin and friends ended up with the following debts:

Person	**Payment details**
Sigge	Paid 2300 SEK for cabin (all share)
Martin	Paid 854 SEK for gasoline (all share)
Martin	Lent Sigge 100 SEK
Anders	Paid 1120 SEK for food (all share)
Gustaf	Paid nothing
Helen	Paid nothing

Now debts are to be repaid: everyone should have paid their share of the common stuff, and interpersonal debts paid.

Here there are several possible problems: optimizing laziness and optimizing costs.

Paying Debts: Laziness

Person	Payment details
Sigge	Paid 2300 SEK for cabin (all share)
Martin	Paid 854 SEK for gasoline (all share)
Martin	Lent Sigge 100 SEK
Anders	Paid 1120 SEK for food (all share)
Gustaf	Paid nothing
Helen	Paid nothing

Let n be the rows in the table and m the number of people. Optimize for the *total number of transfers*. Give a trivial (bad) solution! [pause for discussion]

Paying Debts: Laziness

Person	Payment details
Sigge	Paid 2300 SEK for cabin (all share)
Martin	Paid 854 SEK for gasoline (all share)
Martin	Lent Sigge 100 SEK
Anders	Paid 1120 SEK for food (all share)
Gustaf	Paid nothing
Helen	Paid nothing

Let n be the rows in the table and m the number of people. Optimize for the *total number of transfers*. Give a trivial (bad) solution!

We can just walk down the table and clear debts one by one. For example Martin pays Sigge and then gets paid by Sigge twice. $n(m-1)$ transfers worst case (14 payments here).

Paying Debts: Laziness

Person	Payment details
Sigge	Paid 2300 SEK for cabin (all share)
Martin	Paid 854 SEK for gasoline (all share)
Martin	Lent Sigge 100 SEK
Anders	Paid 1120 SEK for food (all share)
Gustaf	Paid nothing
Helen	Paid nothing

Let n be the rows in the table and m the number of people. Optimize for the *total number of transfers*. Give a trivial (bad) solution!

We can just walk down the table and clear debts one by one. For example Martin pays Sigge and then gets paid by Sigge twice. $n(m-1)$ transfers worst case (14 payments here).

Slightly better we can for each set of two people, e.g. $\{\text{Martin}, \text{Sigge}\}$ calculate the *net debt*, and if it is $\neq 0$ make one payment. $m(m-1)$ worst case (6 payments here).

Paying Debts: Laziness

Person	Payment details
Sigge	Paid 2300 SEK for cabin (all share)
Martin	Paid 854 SEK for gasoline (all share)
Martin	Lent Sigge 100 SEK
Anders	Paid 1120 SEK for food (all share)
Gustaf	Paid nothing
Helen	Paid nothing

Let n be the rows in the table and m the number of people. Optimize for the *total number of transfers*. Give a trivial (bad) solution!

We can just walk down the table and clear debts one by one. For example Martin pays Sigge and then gets paid by Sigge twice. $n(m-1)$ transfers worst case (14 payments here).

Slightly better we can for each set of two people, e.g. $\{\text{Martin}, \text{Sigge}\}$ calculate the *net debt*, and if it is $\neq 0$ make one payment. $m(m-1)$ worst case (6 payments here).

How do we compute perfect solutions though?

Paying Debts: Perfect Laziness

Person	Payment details
Sigge	Paid 2300 SEK for cabin (all share)
Martin	Paid 854 SEK for gasoline (all share)
Martin	Lent Sigge 100 SEK
Anders	Paid 1120 SEK for food (all share)
Gustaf	Paid nothing
Helen	Paid nothing

In reality Gustaf and Helen paid Sigge 854 SEK each, then Sigge paid Anders 265 SEK and Martin 100 SEK, for 4 transfers. Probably minimal (?).

- Note the savings by Gustaf and Helen "overpaying" Sigge who then transferred to Anders. This saved 1 transfer.
- Notice that outside of the 100 SEK loan Martins debt evened out to 0 globally: an important special case.

Solution found ad-hoc, in general? [discuss!]

- Is there always an $m - 1$ solution?
- Is there an instance where $m - 1$ is the minimal solution?

Paying Debts: Other Details

- Bank transfers may have complex costs: assume 2% for transfers of more than 50 SEK, and 1 SEK for all smaller transfers.
- Then this "chaining", where Sigge is overpaid, may be a bad idea!
- In general, to work with this problem the details must be formalized: what are the rules, what is optimized, and *what is the input*?
- The last is important, as the table given previously have both shared and interpersonal debts. Does this matter? Can it be converted to *just interpersonal* without problems? Is it easier with *just shared*?

Lets look at an even trickier case of problem delimiting.

Generalizing and Narrowing Problems, an Example

Above is a game of the excellent board game Ticket to Ride starting out. The *entirety* of the game is much too complex to explain now or to analyze algorithmically.

Generalizing and Narrowing Problems, an Example

The very first step is interesting though:

- Each player draws 3 *route cards* and chooses 2–3 of them.
- A route card lists two cities and a score paid.
- Each chosen card is a committal to connect the two cities by rail.
- Good route selections cause a lot of overlap: reusing rail links

Generalizing a Problem

First we need to generalize: Remove the limit of 2–3 routes (now choose m routes out of n, with $m \leq n$) and the fixed map (now an arbitrary graph).

If there are only *finitely many problem instances* all algorithms are in *constant time*.

This is *not* artificial; the number of problem instances is huge, the finiteness is in practice illusory. This brings the mathematics in line.

Narrowing a Problem

We ignore most of the game, lets consider how to optimize route selection. Ticket to Ride also has different costs for various links, multiple links between the same cities, but lets ignore that too (all links cost 1). This becomes a small heuristic for a part of the game.

Overlapping Route Optimization

- **Input:** A graph $G = (V, E)$, a set $R \subseteq V \times V \times \mathbb{N}$, a constant $k \in \mathbb{N}$.
- **Problem:** Answer "yes" if and only if there exists some $E' \subseteq E$ such that

  $$k \leq \sum\{s \mid (r, r', s) \in R, r \text{ is connected to } r' \text{ in } (V, E')\} - |E'|$$

That is, finding E' corresponds to picking what links to build in G, then we count all routes in R that are fulfilled, and see if the result exceeds k.

This looks hard enough in itself! Ideas? Trivial inefficient solution?

Other Aspects of Ticket to Ride

Notice that dropping route costs (i.e. connecting two cities has an arbitrary integer cost) does not *really* weaken the problem:

- In the game connecting e.g. Nashville and Little Rock "costs" 3
- The "stupid" translation of the game into Overlapping Route Optimization has $\{\text{New York}, \text{Little Rock}\} \subseteq V$ and $(\text{New York}, \text{Little Rock}) \in E$, gives "cost" of 1
- However, Overlapping Route Optimization can deal with integer costs fine, instead let $\{\text{New York}, \text{NYLR1}, \text{NYLR2}, \text{Little Rock}\} \subseteq V$ and $\{(\text{New York}, \text{NYLR1}), (\text{NYLR1}, \text{NYLR2}), (\text{NYLR2}, \text{Little Rock})\} \subseteq E$, where NYLR1 and NYLR2 are new "on the way" nodes not used for anything else.
- In this way the cost is simulated, three edges are needed to connect the nodes corresponding to New York and Little Rock.

The instance grows linearly in the costs represented (be careful!).

Bonus Comment: Even Further Narrowing

Recall Overlapping Route Optimization:

Overlapping Route Optimization

- **Input:** A graph $G = (V, E)$, a set $R \subseteq V \times V \times \mathbb{N}$, a constant $k \in \mathbb{N}$.
- **Problem:** Answer "yes" if and only if there exists some $E' \subseteq E$ such that

  $$k \leq \sum\{s \mid (r, r', s) \in R, r \text{ is connected to } r' \text{ in } (V, E')\} - |E'|$$

If time remains, a final discussion:

- Narrow the problem until you are sure it can be efficiently solved.
- Dropping costs would make it some kind of spanning tree problem where not all nodes need be spanned...

Conclusion

We have seen:

- Information on the way the course works.
- Intuitive descriptions of two problems:
  - Any number of variations and subproblem selections possible
  - Feel free to pick something out of them as your own problem!

Welcome to Algorithmic Problem Solving HT14!