A–Fair Division
B–Free Goodies
C–High Score
D–Hill Driving
E–Rankings
F–Risk
G–Selling Land
H–Stock Prices
I–Telephone Network
J–Wormly

ACM ICPC — Training Session IV

C. Maria Keet

Department of Computer Science, University of Cape Town, South Africa
mkeet@cs.uct.ac.za

September 13, 2014

Today

- NWERC 2010 Regional
- Try to solve as many as possible in 3.5h
- Then we'll discuss the solutions[1] (approach & the code)

[1] The remainder of these slides are based on the Jury's slides, with some additions and rewording for training purposes

Problem A–Fair Division

A–Fair Division

B–Free Goodies

C–High Score

D–Hill Driving

E–Rankings

F–Risk

G–Selling Land

H–Stock Prices

I–Telephone Network

J–Wormly

- Sort persons according to maximum contribution
- Tie-breaker: position in list
- Multiples of 1 cent (integers)
- If dividing according to the rules is not possible (i.e., if `sum` $<$ `P`), then print this
- else: the contribution is `min(max[i], price/(N-i))` for `i=0...N-1`

Problem B–Free Goodies

A–Fair Division
B–Free Goodies
C–High Score
D–Hill Driving
E–Rankings
F–Risk
G–Selling Land
H–Stock Prices
I–Telephone Network
J–Wormly

- Requested: total value of goodies for each
- Petra chooses "most valuable". greedy? "ties": dynamic programming?
- Jan with "multiple choice lead to same optimal result". dynamic programming?
- Sort goodies based on Petra's valuations (why not Jan's?)
- Then popping the queue, and adding up the values

Problem C–High Score

ACM ICPC — Training Session IV
C. Maria Keet
A–Fair Division
B–Free Goodies
C–High Score
D–Hill Driving
E–Rankings
F–Risk
G–Selling Land
H–Stock Prices
I–Telephone Network
J–Wormly

- e.g., start with `AAA` and end with `JAN`
  - first count from `A` to `J` (9 steps), then move to third (1 step to the left), and then `A` to `N` (13 steps).
- For a next letter to update, can we continue where we 'left off'?
  - say, having counted from `A` to `J`, then move to third (1 step to the left), and then `J` to `N`??
- Loops turning around over the name and over the alphabet
- Count number of steps for each character, starting from `A`

Problem C–High Score

ACM ICPC — Training Session IV
C. Maria Keet
A–Fair Division
B–Free Goodies
C–High Score
D–Hill Driving
E–Rankings
F–Risk
G–Selling Land
H–Stock Prices
I–Telephone Network
J–Wormly

- e.g., start with `AAA` and end with `JAN`
  - first count from `A` to `J` (9 steps), then move to third (1 step to the left), and then `A` to `N` (13 steps).
- For a next letter to update, can we continue where we 'left off'?
  - say, having counted from `A` to `J`, then move to third (1 step to the left), and then `J` to `N`??
- Loops turning around over the name and over the alphabet
- Count number of steps for each character, starting from `A`

Problem D–Hill Driving

- Flat land: same speed
- Downhill section: 10 km/h, no fuel consumption
- Uphill section: fuel consumption as if you were driving 10 km/h faster
- Task: calculate *time* it takes to arrive home (≤24h), or impossible.
- Linear search to crunch through the segments and update, or binary search to cut it off earlier if it would be impossible

Problem D–Hill Driving

- Flat land: same speed
- Downhill section: 10 km/h, no fuel consumption
- Uphill section: fuel consumption as if you were driving 10 km/h faster
- Task: calculate *time* it takes to arrive home ($\leq$24h), or impossible.
- Linear search to crunch through the segments and update, or binary search to cut it off earlier if it would be impossible

Problem D–Hill Driving

ACM ICPC — Training Session IV
C. Maria Keet
A–Fair Division
B–Free Goodies
C–High Score
D–Hill Driving
E–Rankings
F–Risk
G–Selling Land
H–Stock Prices
I–Telephone Network
J–Wormly

- Flat land: same speed
- Downhill section: 10 km/h, no fuel consumption
- Uphill section: fuel consumption as if you were driving 10 km/h faster
- Task: calculate *time* it takes to arrive home ($\leq$24h), or impossible.
- Linear search to crunch through the segments and update, or binary search to cut it off earlier if it would be impossible

Problem E–Rankings

A–Fair Division
B–Free Goodies
C–High Score
D–Hill Driving
E–Rankings
F–Risk
G–Selling Land
H–Stock Prices
I–Telephone Network
J–Wormly

- Take last year's ranking as starting point for computing this year's ranking
- Sort swaps along ranking
- Do the swaps
- Check consistency old v.s. new ranking; e.g.,
  - ranking 1 2 3 4
  - swaps 1 2, then 2 3, and finally 3 4
  - results in 2 1 3 4 - 3 1 2 4 - 4 1 2 3
  - 4 1 2 3 violates swap 1 2

Problem F–Risk

- Don't get sidetracked with the rules of the actual game.... (like whether the amount of moves has to match the total number of the dice you throw—that info is not in the input)
- First: find the 'weakest region' bordering an opponent. quickest: binary search
- There is a maximum flow algorithm to determine if the answer is possible

Problem F–Risk

ACM ICPC — Training Session IV
C. Maria Keet
A–Fair Division
B–Free Goodies
C–High Score
D–Hill Driving
E–Rankings
F–Risk
G–Selling Land
H–Stock Prices
I–Telephone Network
J–Wormly

- Don't get sidetracked with the rules of the actual game.... (like whether the amount of moves has to match the total number of the dice you throw—that info is not in the input)
- First: find the 'weakest region' bordering an opponent. quickest: binary search
- There is a maximum flow algorithm to determine if the answer is possible

Problem F–Risk

ACM ICPC — Training Session IV

C. Maria Keet

A–Fair Division

B–Free Goodies

C–High Score

D–Hill Driving

E–Rankings

F–Risk

G–Selling Land

H–Stock Prices

I–Telephone Network

J–Wormly

- Set up graph vertices: the source with the armies, the sink where they have to go to (region bordering opponent's region with least armies), and 2 vertices for each land you control (which text in the description says that?)
- Graph edges:
  - source $\rightarrow$ 1st land (cap is the number of armies)
  - 1st land $\rightarrow$ 2nd land (if connected)
  - 2nd land $\rightarrow$ sink (cap is the needed armies)

Problem G–Selling Land

A–Fair Division
B–Free Goodies
C–High Score
D–Hill Driving
E–Rankings
F–Risk
G–Selling Land
H–Stock Prices
I–Telephone Network
J–Wormly

- Perhaps ‘confusing’ part is not calculating number of cells, but perimeter values
- south-east corner cell no more than once
- Find the number of single cells (perimeter 4)
- Find the number of 2-cells (perimeter 6)
- etc.
- Do that by scanning row-by-row
- For each column c of row r, count the number of grass squares above (c, r)
- Process columns and keep a list of possible end squares

Problem G–Selling Land

- Perhaps 'confusing' part is not calculating number of cells, but perimeter values
- south-east corner cell no more than once
- Find the number of single cells (perimeter 4)
- Find the number of 2-cells (perimeter 6)
- etc.
- Do that by scanning row-by-row
- For each column c of row r, count the number of grass squares above (c, r)
- Process columns and keep a list of possible end squares

Problem H–Stock Prices

- make an array with ask (sell price), bid (buy price), stock (sold), amounts as columns, against the n transaction steps
- process each input line of the test case one at a time:
  - While bid price larger than ask price,
  - process deal up to $min(amount\text{-}ask, amount\text{-}bid)$ number of shares
  - update amounts, stock price
  - output prices or a dash if it doesn't exist, in requested format

I–Telephone Network

ACM ICPC — Training Session IV

C. Maria Keet

A–Fair Division

B–Free Goodies

C–High Score

D–Hill Driving

E–Rankings

F–Risk

G–Selling Land

H–Stock Prices

I–Telephone Network

J–Wormly

- All sets of requests are possible, so we can add dummy requests to get bipartite graph with $deg(v) = 1$ for all v
- Reduce this graph modulo 2^{n-1} to get a bipartite graph with $deg(v) = 2$ for all v
- Split this graph in two graphs with all degrees 1 and you get two instances of the same problem with $n' = n/2$
- Solve recursively and construct solution

J–Wormly

- With $l, b, n \leq 1000000$, would brute force work, calculating all permutations?
  - yes, but probably TLE
  - would it make sense to search the entire search space to check?
- better: use Greedy, moving first leg, then last leg, then the bubbles
- repeat until finished

J–Wormly

ACM ICPC — Training Session IV

C. Maria Keet

A–Fair Division

B–Free Goodies

C–High Score

D–Hill Driving

E–Rankings

F–Risk

G–Selling Land

H–Stock Prices

I–Telephone Network

J–Wormly

- With $l, b, n \leq 1000000$, would brute force work, calculating all permutations?
  - yes, but probably TLE
  - would it make sense to search the entire search space to check?
- better: use Greedy, moving first leg, then last leg, then the bubbles
- repeat until finished

J–Wormly

- With $l, b, n \leq 1000000$, would brute force work, calculating all permutations?
  - yes, but probably TLE
  - would it make sense to search the entire search space to check?
- better: use Greedy, moving first leg, then last leg, then the bubbles
- repeat until finished

Scheduled training dates

- Aug 2: 9:30-15:30
- Aug 16: 9:30-15:30
- Aug 30: 9:30-15:30
- Sept 13: 9:30-15:30
- Sept 26: deadline for registering for the regionals
- **Sept 27: 9:30-15:30**
- Date of the regionals: 4 October, 2014