Bergen Open 2019
Solution Slides

November 2, 2019

UNIVERSITY OF BERGEN

The Jury

- Olav Røthe Bakken
- Torstein Strømme

Special thanks:

- Kattis
- Greg Hamerly (Kattis)
- Kirill Simonov (for testing problems)

Howl

- Problem summary: Give a longer howl than Fenrir. Howl must follow given rules.
- Algorithms:
  - `print("A"*(len(input())) + "WHO")`
  - `print(input() + "O")`

- Runtime: *O(n)*

Author: Torstein Strømme **First solved**: 00:07 **Solved by**: 12 teams

Climbing stairs

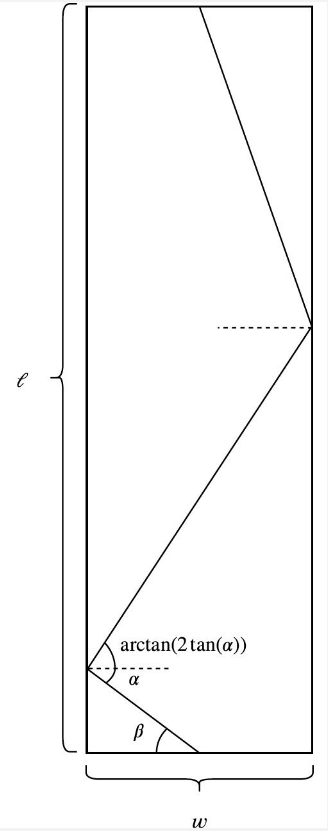
- Problem summary: How many steps are we required to walk each day in order to participate in the staircase cup
- Algorithm:
  - We can always postpone registering to the last possible moment
  - Therefore we will first go to our office, then register at the end of day, then go home
  - If we don't have enough steps when we get to the registration office, pad the number of steps until we have enough, going two steps at a time
  - print (max(n, k + abs(r-k)) + r + (1 if n%2 != r%2 and n > k + abs(k-r) else 0))

- Runtime: *O(1)*

Author: Olav Røthe Bakken **First solved**: 00:34 **Solved by**: 10 teams

Fence bowling

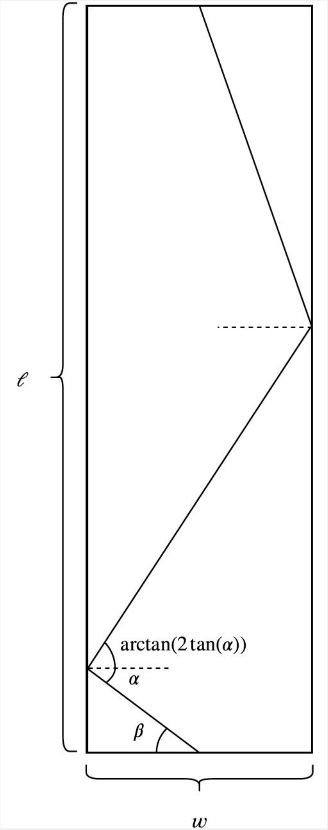
- Problem summary: Determine angle such that you hit strike after *k* bounces.
- Algorithm 1:
  - Binary search on angle β
  - For each guess, simulate bounces
- Runtime: *O(k* log(1/*epsilon*))



Author: Torstein Strømme **First solved**: 01:09 **Solved by**: 4 teams

Fence bowling

- Problem summary: Determine angle such that you hit strike after k bounces.
- Algorithm 2:
  - Observe: the triangle before a bounce is half as “long” (along the centre line) as the triangle after the bounce.
  - There are k pairs of right triangles (following the path from centre line, to side rail, back to centre line), each pair twice as long as the previous pair.
  - Let the first pair of triangles “stretch” a length x. Then total length $L = x + 2x + \dots 2^{k-1}x$
  - Hence, $x = L/(2^k - 1)$
  - Answer is `arctan(L / (2^k - 1) / 3 / (w / 2))`
- Runtime: $O(1)$



Author: Torstein Strømme **First solved**: 01:09 **Solved by**: 4 teams

Bus Ticket

- Problem summary: Decide when to buy single tickets and when to buy period tickets, such that the total cost is minimized.
- Dynamic programming
  - Create array *dp[n]*
  - Define *dp[i]* to be minimum cost to purchase the trips *0…i*
  - Base case: *dp[0]* is price of single ticket (or period ticket, if this is cheaper)
  - Recursive case: *dp[i]* is the minimum of
    - buying a single ticket for the last trip: *dp[i-1]* + price of single trip
    - buying a period ticket for the last trip: *dp[j]* + price of period ticket, where *j* is the latest trip for which a period ticket can not cover both trip *j* and trip *i*.
- Runtime: $O(n^2)$

Author: Torstein Strømme **First solved**: 01:36 **Solved by**: 1 team

Bus Ticket

- Problem summary: Decide when to buy single tickets and when to buy period tickets, such that the total cost is minimized.
- Dynamic programming
  - Create array *dp[n]*
  - Define *dp[i]* to be minimum cost to purchase the trips *0...i*
  - Base case: *dp[0]* is price of single ticket (or period ticket, if this is cheaper)
  - Recursive case: *dp[i]* is the minimum of
    - buying a single ticket for the last trip: *dp[i-1]* + price of single trip
    - buying a period ticket for the last trip: *dp[j]* + price of period ticket, where *j* is the latest trip for which a period ticket can not cover both trip *j* and trip *i*.
- Runtime: ~~$O(n^2)$~~ $O(n \log n)$ (with binary search to find *j*)

Author: Torstein Strømme **First solved**: 01:36 **Solved by**: 1 team

Bus Ticket

- Problem summary: Decide when to buy single tickets and when to buy period tickets, such that the total cost is minimized.
- Dynamic programming
  - Create array *dp[n]*
  - Define *dp[i]* to be minimum cost to purchase the trips *0...i*
  - Base case: *dp[0]* is price of single ticket (or period ticket, if this is cheaper)
  - Recursive case: *dp[i]* is the minimum of
    - buying a single ticket for the last trip: *dp[i-1]* + price of single trip
    - buying a period ticket for the last trip: *dp[j]* + price of period ticket, where *j* is the latest trip for which a period ticket can not cover both trip *j* and trip *i*.
- Runtime: ~~$O(n^2)$~~ ~~$O(n \log n)$~~ $O(n)$ (with sliding pointer to find *j*)

Author: Torstein Strømme **First solved**: 01:36 **Solved by**: 1 team

Alehouse

- Problem summary: During which time interval of length *k* can you meet the most different people in the alehouse?
- What if interval has length 0?
  - For each person, make two events: Arrival and departure.
  - Sort all events (sort arrivals before departures)
  - count = 0
  - for each event in events:
    - if event is arrival, count++
    - if event is departure, count--
  - Remember maximum value of count

Author: Torstein Strømme **First solved**: 01:45 **Solved by**: 1 team

Alehouse

- Problem summary: During which time interval of length k can you meet the most different people in the alehouse?
- What if interval has length 0?
    - Can solve in time $O(n \log n)$
- Observation:
    - You stay for k seconds $\Leftrightarrow$ You stay for 0 seconds, everyone else stays for k seconds longer

- Runtime: $O(n \log n)$

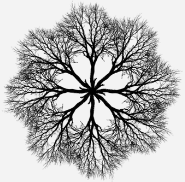
Author: Torstein Strømme **First solved**: 01:45 **Solved by**: 1 team

Great GDP

- Problem summary: Find the connected subtree containing the root with the largest gdp per capita
- Algorithm:
  - If the root has the largest gdp per capita we can simply select the root
  - Otherwise some other vertex v has largest gdp per capita
  - Every solution which includes v must also include parent[v]
  - Merge v and parent[v]
    - Can use union-find to keep track of gdp, population and parent
  - Use a priority queue to quickly find the vertex with highest gdp per capita
- Runtime: $O(n \log n)$

Author: Olav Røthe Bakken **First solved**: 02:07 **Solved by**: 1 team

Equilibrium

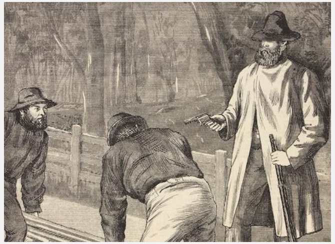


- Problem summary: Find the order of vertices which minimizes imbalance
- Algorithm:
  - There exists an ordering where every vertex with even degree has imbalance 0, and every vertex with odd degree has imbalance 1
  - Pick a vertex as the root, and distribute its children evenly on either side
  - Disjoint subtrees will not interfere with each other, so we can assume the vertices from each subtree are contiguous in the optimal ordering
  - Recursively find the order of each subtree

- Runtime: $O(n)$

Author: Olav Røthe Bakken **First solved**: 02:31 **Solved by**: 1 team

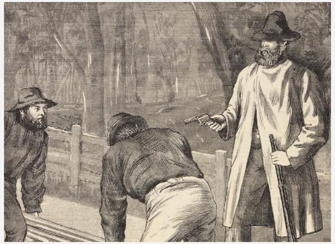
Killing Chaos



- ➢ Problem summary: Figure out the maximum chaos according to the rules
- ➢ Rules: chaos = # of train segments * sum(round up to closest 10 the # of passengers in each segment)
- ➢ Naive algorithm:
  - Simulate the process
  - Keep an array which keeps track of whether each wagon is killed
  - Each time a wagon is blown up, recalculate the chaos
- ➢ Runtime: $O(n^2)$

Author: Torstein Strømme **First solved**: 03:14 **Solved by**: 1 team

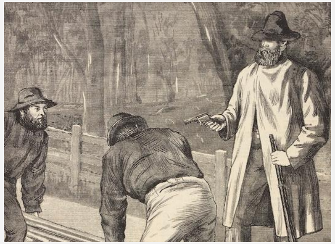
Killing Chaos



- Problem summary: Figure out the maximum chaos according to the rules
- Rules: chaos = # of train segments * sum(round up to closest 10 of passengers in each segment)
- Better algorithm:
  - Simulate the process *backwards*
  - Use union-find to keep track of how many passengers in each segment
  - Keep track of number of segments, and the “base chaos” (before multiplication with number of segments)
- Runtime: $O(n \log^* n)$

Author: Torstein Strømme **First solved**: 03:14 **Solved by**: 1 team

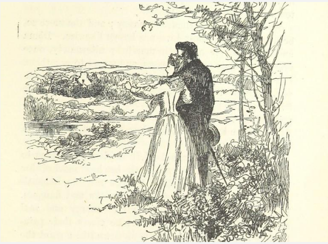
Killing Chaos



- Problem summary: Figure out the maximum chaos according to the rules
- Rules: chaos = # of train segments * sum(round up to closest 10 of passengers in each segment)
- Another good algorithm:
  - Keep a sorted set (binary search tree) which contains train segments (lower bound, upper bound, # of people)
  - Keep track of number of segments, and the "base chaos" (before multiplication with number of segments)
  - When a coach is killed, remove corresponding segment from sorted set (found in log(n) time), and add back smaller segments if necessary.
- Runtime: $O(n \log n)$

Author: Torstein Strømme **First solved**: 03:14 **Solved by**: 1 team

Jane Eyre

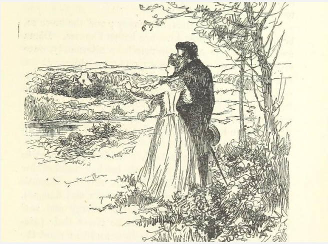


- Problem summary: Given that Anna always reads in her books in alphabetical ASCII order, when will she (at the earliest) finish reading Jane Eyre? Books arrive as time goes.
- Simulation
  - Let time be 0
  - Pick the earliest book from priority queue sorted by ASCII order; read it and update time
  - Receive all new books that arrive at current time or earlier, put those in priority queue (use sliding pointer)
  - Repeat until Jane Eyre is read

- Runtime: $O(n \log n)$

Author: Torstein Strømme **First solved**: 03:34 **Solved by**: 4 teams

Jane Eyre



- Problem summary: Given that Anna always reads in her books in alphabetical ASCII order, when will she (at the earliest) finish reading Jane Eyre? Books arrive as time goes.
- Alternative simulation
  - Ignore all books after Jane Eyre in ASCII alphabet
  - Sort books by arrival time
  - Read the books, track the time; continue until the next book arrives after the current time
  - Return current time + time to read Jane Eyre

- Runtime: $O(n \log n)$

Author: Torstein Strømme **First solved**: 03:34 **Solved by**: 4 teams

Ice cream

- Problem summary: Produce as much chocolate ice cream as possible
- Algorithm:
  - We want to compute the maximum amount of flow (W) from c and v to f, such that the flow from the chocolate tank c is equal to the flow from the vanilla tank v.
  - Convert into a standard max flow problem by binary search for the answer
    - Add a super-source with pipes to c and v that each have capacity g (half the guessed flow)
    - It is possible the optimal solution uses half integral amounts of each ingredient
  - Implement using your favourite max-flow algorithm (e. g. Edmund's Karp)

- Runtime: $O(nm^2 \log W)$

Author: Olav Røthe Bakken | **First solved**: - | **Solved by**: 0 teams

Drive safely

- Problem summary: Given a polyline describing a road, place speed signs such that travel time by travelling legally is as small as possible.
- Some basic geometry to find angles and distances
- Dynamic programming:
  - Two tables: dp_a[n][k] and dp_b[n][k]
  - Define dp_a[i][j] = Minimum time required to travel to (just before) point i using j or less speed signs
  - Define dp_b[i][j] = Minimum time required to travel to (just after) point i using j or less speed signs
  - At location i, check every possible location for the previous speed sign

- Runtime: $O(n^2k)$

Author: Torstein Strømme | **First solved**: - | **Solved by**: 0 teams

Statistics

- Number of teams: 12
- Number of participants: 30
- Number of submissions: 180
- Number of accepted submissions: 35
- First accepted submission: 00:07:54 (Howl)
- Last accepted submission: 04:51:02 (Jane Eyre)
- Number of commits to problem repository: 164

Copyright notes

➢ Pictures
  - Alehouse https://commons.wikimedia.org/wiki/File:Tavern_Scene-1658-David_Teniers_II.jpg (Public Domain)
  - Bus Ticket https://pixabay.com/photos/bus-dublin-ireland-public-transport-2616074/ (Pixabay License, TuendeBede)
  - Climbing Stairs https://commons.wikimedia.org/wiki/File:السلم الداخلى لبرج قصر البارون.jpg (CC-BY-SA 4.0, Manadily)
  - Drive Safely https://pxhere.com/en/photo/499914 (CC0)
  - Equilibrium https://pixabay.com/illustrations/tree-kahl-ring-wreath-branch-2938948/ (Pixabay License, Gerd Altmann)
  - Great GDP https://pixabay.com/photos/tree-building-city-park-green-life-1333428/ (Pixabay License, Timofey Iasinskii)
  - Howl https://pixabay.com/illustrations/wolf-howl-moon-weird-surreal-963081/ (Pixabay License, Alexas Fotos)
  - Ice Cream https://www.yokota.af.mil/News/Photos/igphoto/2002083858/mediaid/3167207/ (Public Domain, U.S. Air Force)
  - Jane Eyre https://www.flickr.com/photos/britishlibrary/11298156955 (Public Domain, British Library)
  - Killing Chaos https://commons.wikimedia.org/wiki/File:Ned_Kelly_attemps_to_derail_train.jpg (Public Domain, The Australian Pictorial Weekly)