The Jury

➢ Olav Røthe Bakken
➢ Petter Andre Dahl Elvevoll
➢ Øyvind Stette Haarberg
➢ Torstein Strømme
➢ Birk Tjelmeland
➢ Erik Tjøswold
➢ Amar Topalovic
➢ Magnus Øian
License to Launch

➢ Problem summary: Find first occurrence of smallest value in list of numbers.

➢ Algorithm:
  ○ Check each value in the provided order
  ○ Keep a running minimum value and remember its index
  ○ Print the remembered index when loop is finished

➢ One-liner (python3):

```python
print(min((v, i) for i, v in enumerate(int(x) for x in input().split()))[1] if input() else "")
```

➢ Runtime: $O(n)$
Fishmongers

➢ Problem summary: Sell fish to make as much monies as possible.
➢ Algorithm:
  ○ Sort your fish w.r.t weight
  ○ Sort fishmongers w.r.t price
  ○ Sell your biggest fish to the buyer who is willing to pay the most, until either no more fish or no more buyers
➢ Runtime: $O(n \log n)$
Problem summary: Given a list of integers, find the shortest distance between any pair of equal integers.

Algorithm:
- Maintain a dictionary which maps each integer to its previously seen position.
- Keep track of the shortest distance $d$.
- For each integer in the list:
  - If integer is encountered previously, check if difference between previous and current position is less than $d$ and update accordingly.
  - Update previous position of the given integer.

Runtime: $O(n)$
Joint Attack

➢ Problem summary: Given a number $x$ as a continued fraction, output $x$ as a reduced fraction.

➢ Algorithm:
  ○ Keep track of numerator ($\text{num}$) and denominator ($\text{den}$) for each layer.
  ○ Starting at the bottom-most layer:
    ■ Find denominator and numerator of integer plus fraction.
    ■ Reciprocate (switch $\text{num}$ and $\text{den}$) to eliminate a layer.
    ■ Simplify fraction.
    ■ Repeat until only one fraction left.
  ○ Print remaining fraction ($\text{den} + \"/\" + \text{num}$).

\[
x = x_0 + \frac{1}{x_1 + \frac{1}{x_2}}
\]

\[
x = x_0 + \frac{x_2}{x_1 x_2 + 1}
\]

\[
x = \frac{x_0 + x_2 + x_0 x_1 x_2}{x_1 x_2 + 1}
\]
Counting Clauses

➢ Problem summary: Determine whether a given SAT formulae has eight clauses or more.

➢ Algorithm:
  ○ Read first number of input
  ○ If that number is ≥ 8, print “satisfactory”
  ○ Otherwise print “unsatisfactory”

➢ One-liner (python3):

```python
print("satisfactory" if int(input().split()[0]) >= 8 else "unsatisfactory")
```

Author: Øyvind Stette Haarberg
First solved: 00:36
Solved by: 26 teams
Keyboards in Concert

➢ Problem summary: Play a tune with instruments, each with limited access to notes. Switch instrument as little as possible.

➢ Algorithm:
  ○ Observe that we want to start with the instrument which gets us the farthest
  ○ Don’t need to compute this; simply maintain list of valid instruments after each note
  ○ When no more valid instruments: Increment a counter and reset group of valid instruments

➢ Runtime: $O(mn + nk)$

Author: Olav Røthe Bakken  First solved: 01:18  Solved by: 7 teams
Backpack Buddies

➢ Problem summary: A race between two players in a weighted graph; Mr. Day is required to end each 12 hour increment at a vertex, whereas Dr. Knight need not.

➢ Algorithm:
  ○ Run normal Dijkstra to determine walking time required for Dr. Knight. Compute time she spent resting.
  ○ Run a special Dijkstra for Mr. Day where he ends every day at a vertex
    ■ Assume Mr. Day arrives at vertex \( u \) at day \( d \) and hour \( h \)
    ■ Assume there is an edge from \( u \) to neighbour \( v \) which takes \( w \) hours to traverse. Then:
      ● If \( h + w \leq 12 \), it is possible to arrive at \( v \) at day \( d \) and hour \( h + w \)
      ● Otherwise, it is possible to arrive at \( v \) at day \( d + 1 \) and hour \( w \)

➢ Runtime: \( O(m \log n) \)
Hidden Words

Problem summary: Given a grid of letters and a list of words, count the number of words in the list that occur in the grid

Algorithm:

- First construct every possible word in the grid:
  - Start from every cell: Run DFS with max “depth” 10 that unmarks cell as visited after visiting neighbors
  - While in the DFS, construct a trie from the found words
  - Observe: (significantly) less than $\sum_{i=0..9} h \cdot w \cdot 4 \cdot 3^{i-1} \approx 3 \cdot 10^6$ such words.
- For every word in the list, increment a counter if it exists in the trie

Runtime: $\sim 4 \cdot 3^8 h w + 10n$
Hidden Words

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

Words in Trie:
- A
- AB
- ABA
- ABAC
- AC
- ACA
- ACAB
- B
- BA
- BAC
- BACA
- C
- CA
- CAB
- CABA

Author: Amar Topalovic

First solved: 02:43
Solved by: 3 teams
Expecting Rain

- **Problem summary:** Get to the bus as dry as possible within time limit.
- **Algorithm:**
  - Define $dp[i][j]$ as the minimum amount of rain you can expect being at position $i$ at time $j$
  - Want to find $dp[d][t]$
  - Base case observations:
    - $dp[0][j] = 0$ (since there is always roof at distance 0 from home)
    - $dp[i][0] = \infty$ (unless $i = 0$) (since we must start from home)
  - Observe: Can assume all waiting happens at roof endpoints (notable exception: the bus stop itself, if roofed)
  - Recurrence:
    - If roof endpoint at position $i$, then $dp[i][j] = \min(dp[i][j-1], dp[i-1][j-1])$
    - If no roof at position $i$, then $dp[i][j] = dp[i-1][j-1] + \text{expected rain at that time interval}$
Expecting Rain

➢ Problem summary: Get to the bus as dry as possible within time limit.
➢ Algorithm:
  ○ Preprocessing: For each time unit, calculate the expected amount of rain using prefix sum
  ○ Calculate recurrence using dp table or memoization
  ○ Be careful with edge cases

➢ Runtime: $O(c + dt)$
  ○ $d =$ distance to the bus stop, $t =$ time until the bus leaves, $c =$ number of clouds
Problem summary: Given a board of ladder game, find the smallest number $x$, such that after $x$ dice rolls you win the game with probability at least $p$

Background:

- Define $P(x)$ to be true if you can win the game with probability at least $p$ with $x$ dice rolls, false otherwise
- $P$ has the property required for binary search: if $P(x)$ then also $P(x+1)$
- Define $M_i[s, t]$ as the probability of moving from cell $s$, to cell $t$ in $i$ rounds
- Want to find smallest $x$ such that $M_x[1, c\cdot r] \geq p$
- Observe: Matrix $M_1$ can be constructed by examining input
- Observe: $M_i[s, t] = \Sigma_k (M_{i-1}[s, k] \cdot M_1[k, t])$, which implies $M_i = M_{i-1} \times M_1$ and thus $M_i = (M_1)^i$
Dice and Ladders

➢ Algorithm:
  ○ Construct matrix \( M_1 \)
  ○ Binary search on the number of dice rolls, \( x \):
    ■ Compute \( M_x = (M_1)^x \)
    ■ If \( M_x[1, c\cdot r] \geq p \) try smaller \( x \), else try larger \( x \)

➢ Runtime complexity:
  ○ TLE with naive matrix exponentiation
  ○ AC with fast matrix exponentiation: \( a^b = \left( a^{\frac{b}{2}} \right)^2 \)
  ○ Final complexity: \( O((c \cdot r)^3 \log^2 x) \)
  ○ Can be done more cleverly in \( O((c \cdot r)^3 \log x) \)

Author: Birk Tjelmeland
First solved: N/A
Solved by: 0 teams
ISP Merger

➢ Problem summary: make a graph connected with at most k edge additions or deletions, without violating degree constraints

➢ Structural insights:
  ○ When we connect two components we add an edge between one vertex with open connection sockets in component 1, and a vertex with open connection sockets in components 2.
  ○ If we have a component with no free connection spots, we must delete an edge to obtain free spots to connect to other components.
  ○ We don’t care about the size of each connected component or how a connected component is connected except for two details: the number of free connection spots, and the number of removable edges (we can calculate this number by seeing how many more edges than a tree this component has; i.e. since trees have n-1 edges, a component with m edges will have m-(n-1) removable edges.)
ISP Merger

➢ Structural insights (cont’d):
  ○ We only care about the free spots and number removable edges for all connected components
  ○ We want to connect the components with the most free spots first to minimize deletions
  ○ Trees with no free connection spots and trees with 1 free connection spot cause difficulties
  ○ We can’t connect a tree with no free connection spots to another component without violating a degree constraint. We must therefore split all such trees, obtaining two trees with one free connection spot
  ○ When we connect two trees with one free connection spot together we will obtain a tree with no connection spots. We only want to do this as a last step, as the resulting component can’t be connected further.
ISP Merger

➢ Algorithm:
  ○ Find the number of free connection spots and removable edges for each component (find only 1 → “yes”)
  ○ Split up all components which are trees with no connection spots
  ○ While we have more than one component and $k \geq 0$:
    ■ Take the two components with the most free connection spots (exception: trees with one connection spot are sorted last)
    ■ Make sure they have at least free connection spot (delete a non-bridge edge if not -- if no such edge and no free spot output “no”)
    ■ Connect them together
  ○ Make sure to update $k$ for every edit

➢ Runtime (with a priority queue for ordering components): $O (|V| \log |V| + |E|)$
Gameworld Tornado

➢ Problem summary: Compute area of rectangles
➢ Algorithm
  ○ Convert rectangles into events
    ■ Event consists of an x coordinate, and a segment (two y coordinates) and a delta value (+1 or -1)
  ○ Sort events by x
  ○ For each event
    ■ Add \((x - \text{lastx}) \cdot \text{“score” of segment tree root to total area}\)
    ■ Add event delta to segment tree within event delta
    ■ Update lastx
  ○ Output total area
Gameworld Tornado

➢ Segment tree
  ○ Each node contains a value and score
  ○ Value is number of rectangles contained within the segment
  ○ Score is the length covered within the segment
  ○ If value is positive then score is the size of the segment - else score is the sum of the scores of the children

➢ Runtime: $O(n \log n)$

Author: Magnus Øian and Olav Røthe Bakken  
First solved: N/A  
Solved by: 0 teams
Statistics

- Number of teams: 33
- Number of participants: 77
- Number of submissions: 463
- Number of accepted submissions: 115
- First accepted submission: 00:07:25 (License to Launch)
- Last accepted submission: 04:58:23 (Counting Clauses)
- Number of commits to problem repository: 489
Copyright notes

➢ The problems, solution slides, and other materials produced for Bergen Open 2018 are released under CC-BY-SA 4.0.

➢ Pictures

- Gameworld Tornado [https://commons.wikimedia.org/wiki/File:LootHunter_Tileset.png](https://commons.wikimedia.org/wiki/File:LootHunter_Tileset.png) (CC-BY, DragonDePlatino)
- ISP Merger [https://pxhere.com/cs/photo/645980](https://pxhere.com/cs/photo/645980) (CC0, pxhere)
- Joint Attack [https://www.nasa.gov/centers/kennedy/launchingrockets/launchwindows.html](https://www.nasa.gov/centers/kennedy/launchingrockets/launchwindows.html) (Public Domain)
- Keyboards in Concert [https://www.flickr.com/photos/danielspils/25205507](https://www.flickr.com/photos/danielspils/25205507) (CC-BY, Daniel Spils)
- License to Launch [https://commons.wikimedia.org/wiki/File:Falcon_Heavy_clearing_the_tower_04.jpg](https://commons.wikimedia.org/wiki/File:Falcon_Heavy_clearing_the_tower_04.jpg) (Public Domain)