The 2009 ACM North Western European Regional Contest Friedrich-Alexander-University, Nuremberg, Germany

NWERC Jury

November 8, 2009

jury sample solutions

Problem	**min. LOC**	**max. LOC**
An Industrial Spy	24	102
Common Subexpression Elimination	52	133
Divisible Subsequences	14	29
Fractal	36	159
Mountain Road	24	113
Moving to Nuremberg	67	115
Room Assignments	59	110
Settlers of Catan	32	137
Simple Polygon	35	134
Wormholes	56	129
$\sum$	399	1161

An Industrial Spy

- generate all possible numbers
- use backtracking or `next_permutation`
- test primality by trial division or sieve of eratosthenes

Common Subexpression Elimination

- parse expression (recursive descent)
- find equal subtrees

Common Subexpression Elimination

- parse expression (recursive descent)
- find equal subtrees
- comparing whole subtrees (top-down) is too slow
- construct shared trees bottom-up

Common Subexpression Elimination

- parse expression (recursive descent)
- find equal subtrees
- comparing whole subtrees (top-down) is too slow
- construct shared trees bottom-up
- store the first occurrence of a subtree in a table
- identified by label and unique numbers of children
- comparing trees needs just one table lookup

Common Subexpression Elimination

- parse expression (recursive descent)
- find equal subtrees
- comparing whole subtrees (top-down) is too slow
- construct shared trees bottom-up
- store the first occurrence of a subtree in a table
- identified by label and unique numbers of children
- comparing trees needs just one table lookup
- time $O(n \cdot \log n)$ and space $O(n)$

Divisible Subsequences

- naive $O(n^2)$ solution times out
- instead, compute partial sums modulo d
- if two partial sums have the same remainder, their difference is divisible by d
- for each remainder, save the number of corresponding partial sums
- time $O(n + d)$, space $O(d)$

Fractal

- Basic idea: iterate over the depth and find out where you end up.

Fractal

- Basic idea: iterate over the depth and find out where you end up.
- For each depth, iterate over the line segments until you pass the fraction f.
- Then rotate and scale your basis and proceed with the next depth.

Fractal

- Basic idea: iterate over the depth and find out where you end up.
- For each depth, iterate over the line segments until you pass the fraction f.
- Then rotate and scale your basis and proceed with the next depth.
- Note: using complex numbers (e.g. C++'s `complex<double>`) is convenient to represent coordinates for scaling/rotating.
- time $O(n \cdot d)$

Mountain Road

- Sort incoming cars into two lists: left-goers and right-goers
- dynamic programming: find the optimal time when a left-goers and b right-goers have passed and the last car was of type A or B
- try to send $1, 2, 3, \ldots$ cars at a time
- time $O(n^3)$, space $O(n^2)$

Moving to Nuremberg

- For each v, want sum of distances $D(v)$ from v to every other node (weighted by their frequencies)
- Easy to compute contribution to $D(v)$ from nodes in subtree rooted at v
- Find formula for remaining part in terms of $D(\text{parent})$
- Propagate down from the root
- greedy optimisation of convex function
- time $O(n)$

Room assignments

- bipartite graph: persons vs rooms
- valid room assignment is perfect matching

Room assignments

- bipartite graph: persons vs rooms
- valid room assignment is perfect matching
- for each person: connect his two rooms directly
- for each connected component of size m, the number of edges must be $\leq m$
- each connected component must contain at most one cycle

Room assignments

- bipartite graph: persons vs rooms
- valid room assignment is perfect matching
- for each person: connect his two rooms directly
- for each connected component of size m, the number of edges must be $\leq m$
- each connected component must contain at most one cycle
- one tree component, additional components with exactly one cycle

Room assignments

- Distinguish 3 cases:
  1. one component with more than cycle: impossible
  2. only one component: choose the two nodes with highest rating
  3. more components: choose room with highest rating from tree component
     second room either from tree (same rating) or from cyclic component
- time $O(n)$, space $O(n)$

Settlers of Catan

- Nothing really clever, just simulate the procedure
- time $O(n)$ using precomputation

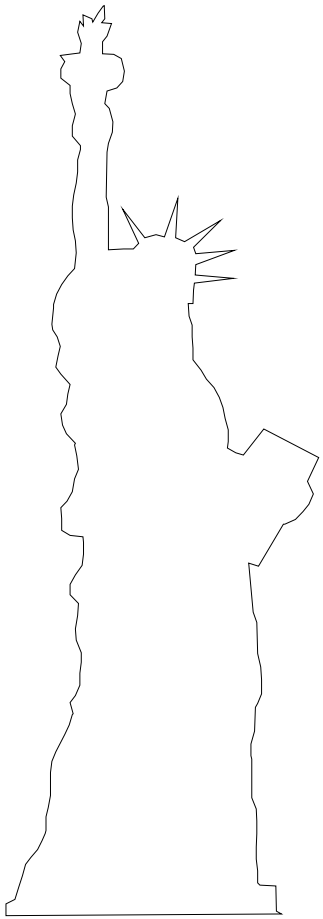
Settlers of Catan

- Nothing really clever, just simulate the procedure
- time $O(n)$ using precomputation
- How to represent a hexagonal lattice?
- How to simulate the spiral?

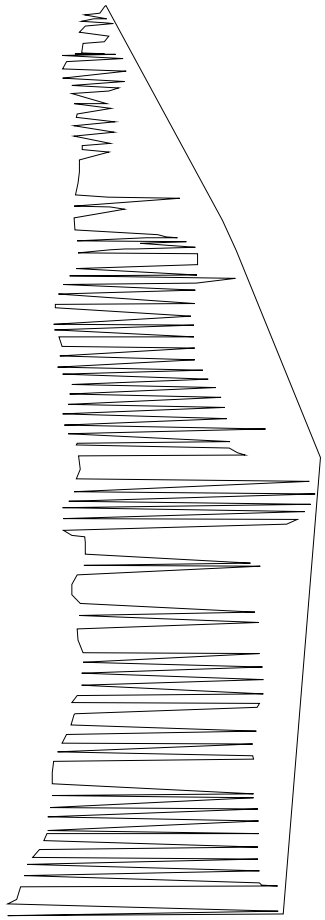
Settlers of Catan

- Nothing really clever, just simulate the procedure
- time $O(n)$ using precomputation
- How to represent a hexagonal lattice?
- How to simulate the spiral?
- Good exercise for solving ad hoc problems and finding easy to code representations.

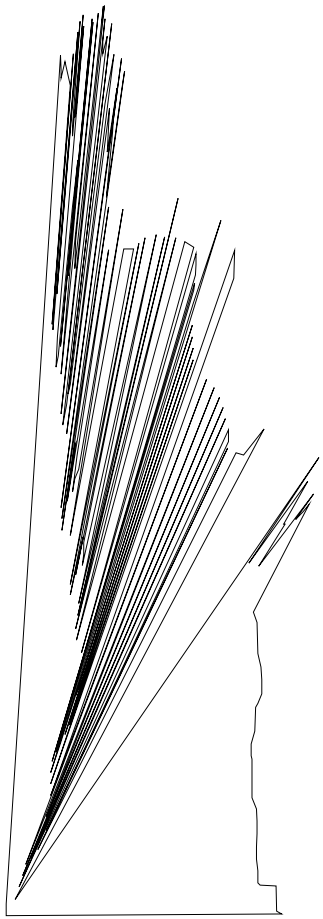
Simple Polygon



Simple Polygon



Simple Polygon



Wormholes

A shortest path problem with negative cycles, but relaxation along edges until ready takes too long.

Wormholes

A shortest path problem with negative cycles, but relaxation along edges until ready takes too long.

- Use the Bellman–Ford algorithm to detect a negative cycle.

Wormholes

A shortest path problem with negative cycles, but relaxation along edges until ready takes too long.

- Use the Bellman–Ford algorithm to detect a negative cycle.
- Recover a cycle by keeping track of by which edge each vertex was last updated.

Wormholes

A shortest path problem with negative cycles, but relaxation along edges until ready takes too long.

- Use the Bellman–Ford algorithm to detect a negative cycle.
- Recover a cycle by keeping track of by which edge each vertex was last updated.
- Find the earliest possible time to traverse this cycle and update vertices.

Wormholes

A shortest path problem with negative cycles, but relaxation along edges until ready takes too long.

- Use the Bellman–Ford algorithm to detect a negative cycle.
- Recover a cycle by keeping track of by which edge each vertex was last updated.
- Find the earliest possible time to traverse this cycle and update vertices.
- Repeat until no more cycles are present.

Wormholes

A shortest path problem with negative cycles, but relaxation along edges until ready takes too long.

- Use the Bellman–Ford algorithm to detect a negative cycle.
- Recover a cycle by keeping track of by which edge each vertex was last updated.
- Find the earliest possible time to traverse this cycle and update vertices.
- Repeat until no more cycles are present.

This gives an $\mathcal{O}(n^4)$ algorithm.

Award Ceremony