Benelux
Algorithm
Programming
Contest
2009

The Results

Business teams

Business teams

VORtech

3 correct submissions,

674 minutes

Third place

Third place

assert (rank == 1)

6 correct submissions,
1387 minutes

Second place

Second place

Joy

6 correct submissions,

(2 minutes before the end)

910 minutes

First place

First place

Doeke en Jelle

7 correct submissions,
1318 minutes

Special Prize

For the shortest and first solution of problem I

Special Prize

For the shortest and first solution of problem I

Joy

Thanks

Thanks

- Runners

Thanks

- Runners
- Fotocie

Thanks

- Runners
- Fotocie
- System administrators

Thanks

- Runners
- Fotocie
- System administrators
- Chipcie

Benelux
Algorithm
Programming
Contest
2009

The Solutions

I. Arctic Polar Explorer

I. Arctic Polar Explorer

- Bubble Sort

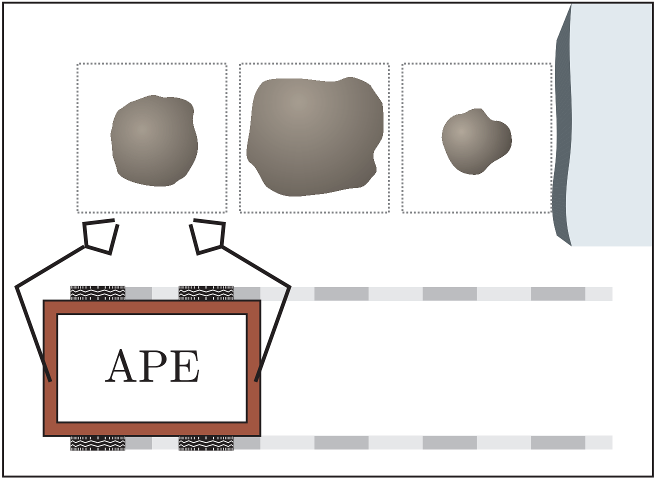
I. Arctic Polar Explorer

- Bubble Sort
- or Gnome Sort

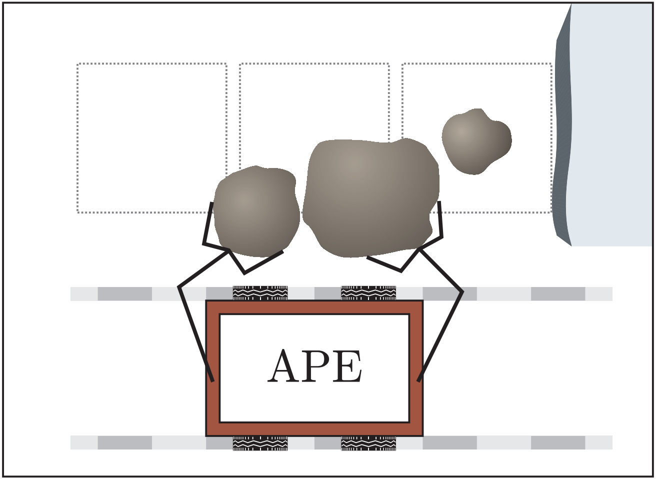
I. Arctic Polar Explorer

- Bubble Sort
- or Gnome Sort
  - Look at two items *a* and *b*.
  - If $a < b$ move right.
  - If $a > b$ swap and move left.
  - At the first rock move right.
  - At the last rock we are done.

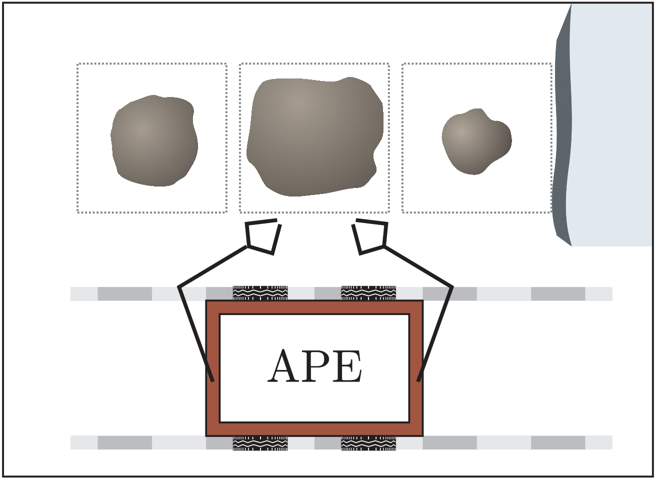
I. Arctic Polar Explorer



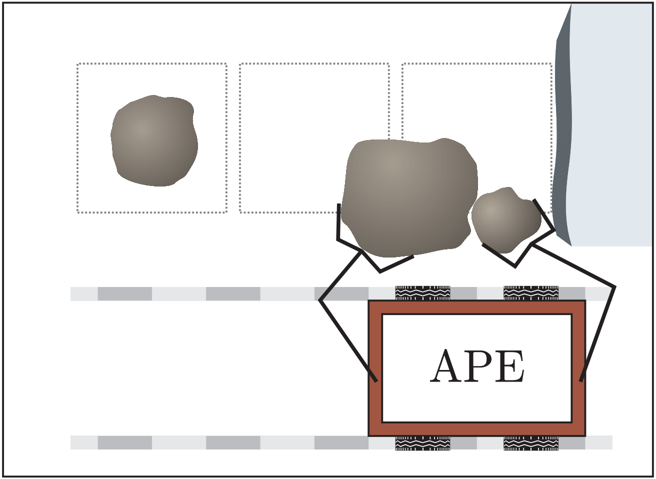
I. Arctic Polar Explorer



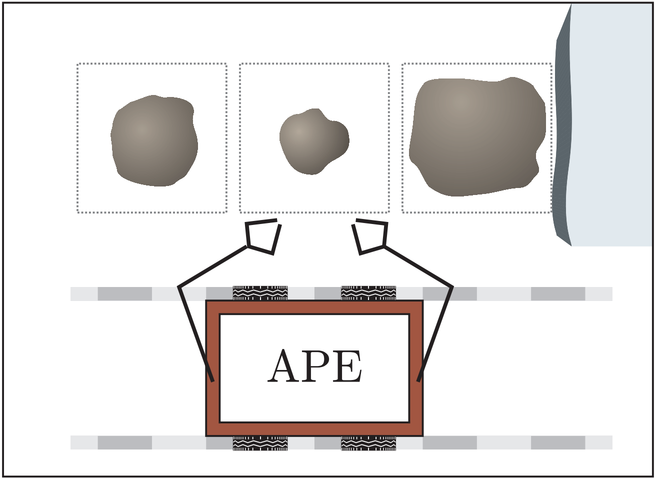
I. Arctic Polar Explorer



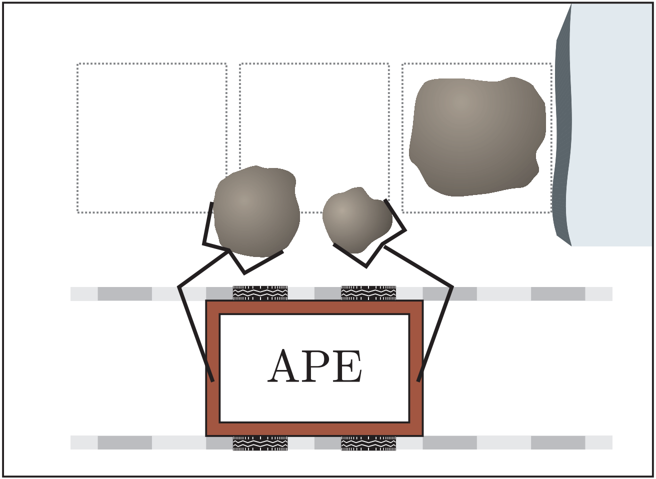
I. Arctic Polar Explorer



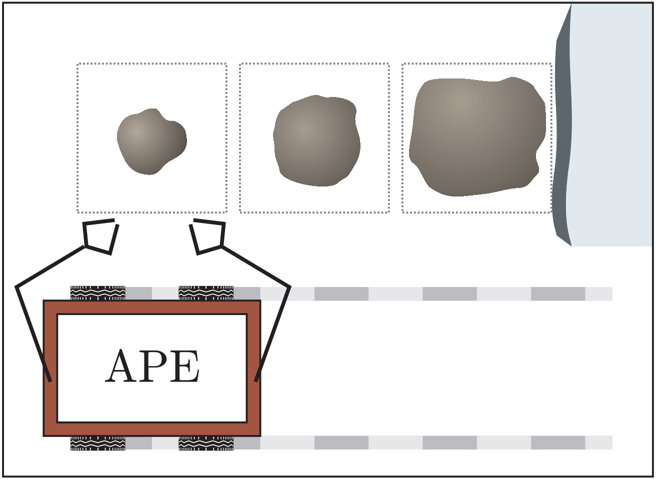
I. Arctic Polar Explorer



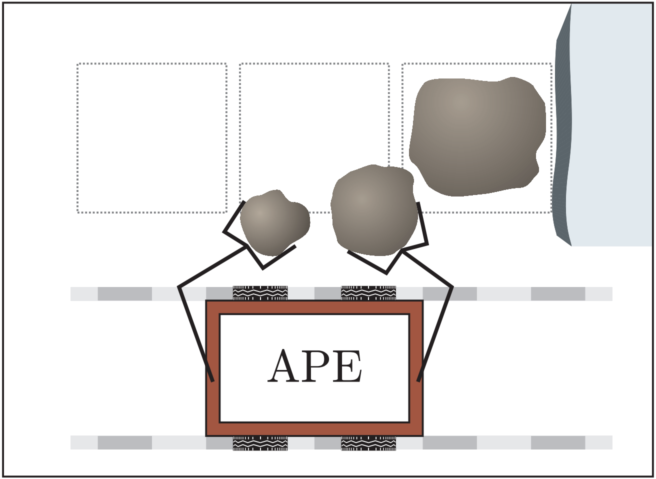
I. Arctic Polar Explorer



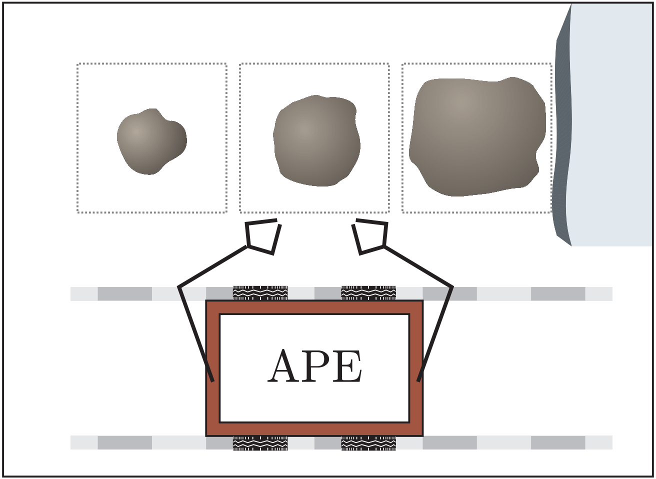
I. Arctic Polar Explorer



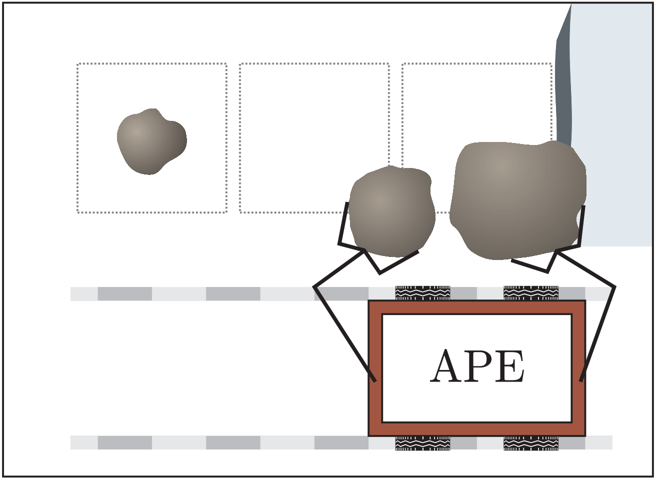
I. Arctic Polar Explorer



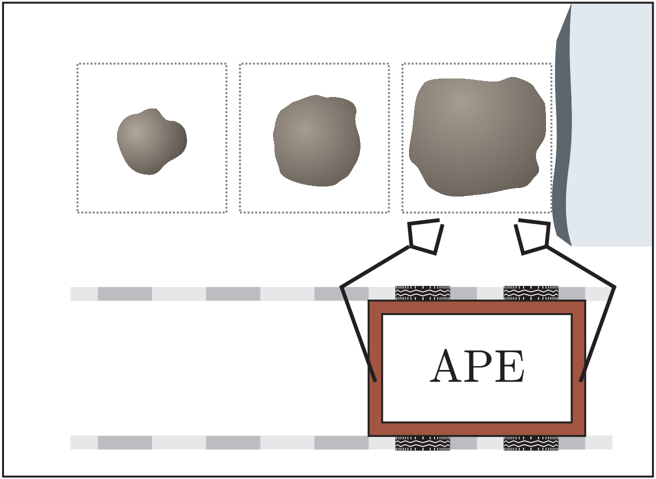
I. Arctic Polar Explorer



I. Arctic Polar Explorer



I. Arctic Polar Explorer



D. Baby’s Blocks

D. Baby's Blocks

- $\text{findIndex} \circ \text{unique} \circ \text{permutations}$

D. Baby's Blocks

- findIndex $\circ$ unique $\circ$ permutations
- Want $\mathcal{O}(n^k)$ instead of $\mathcal{O}(n!)$

D. Baby's Blocks

- findIndex ∘ unique ∘ permutations
- Want $\mathcal{O}(n^k)$ instead of $\mathcal{O}(n!)$
- Easy without duplicates:

$$\sum_i \#\text{smaller after} \cdot (n-i)!$$

D. Baby's Blocks

- findIndex ∘ unique ∘ permutations
- Want $\mathcal{O}(n^k)$ instead of $\mathcal{O}(n!)$
- Easy without duplicates:

$$\sum_i \#\text{smaller after} \cdot (n - i)!$$

- Trickier with duplicates:
- Use multinomial coefficients.

G. Fractal Streets

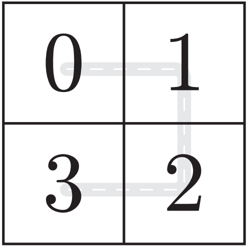
- Hilbert curve

G. Fractal Streets

- Hilbert curve
- Start with least significant two bits

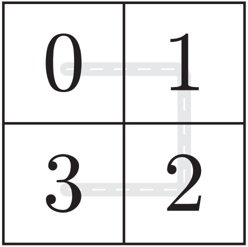
G. Fractal Streets

- Hilbert curve
- Start with least significant two bits



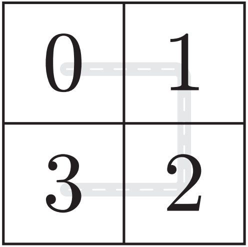
G. Fractal Streets

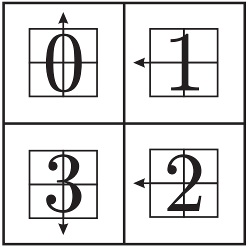
- Hilbert curve
- Start with least significant two bits
- Recursively process higher bits



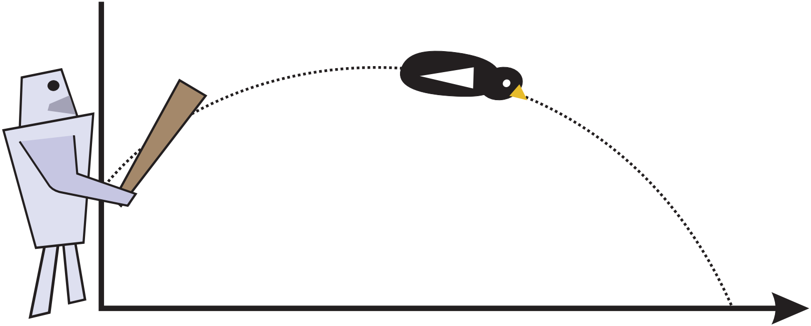
G. Fractal Streets

- Hilbert curve
- Start with least significant two bits
- Recursively process higher bits





A. Penguin Bashing



A. Penguin Bashing

- Simulate for each angle, pick the best one

A. Penguin Bashing

- Simulate for each angle, pick the best one
- First landing point

$$y_0 + y_0' t - {}^1\!/\!_2 G t^2 = 0 \qquad x_t = t x_0'$$

- Slide until next obstacle

$$x_o' = x' - (x - x_o)/5$$

- When hitting a mine, fly forward, skipping obstacles
- Finally slide until you stop.

F. Dimensional Warp Drive

F. Dimensional Warp Drive

- Given *n* vectors of integers modulo 11
- Write target as a linear combination

F. Dimensional Warp Drive

- Given *n* vectors of integers modulo 11
- Write target as a linear combination
- Modified Gaussian elimination

F. Dimensional Warp Drive

target	3	6	0	5	
	1	6	5	5	3079
	1	0	0	7	3075
	0	6	2	5	3078
	4	6	0	1	3082

F. Dimensional Warp Drive

target	3	6	0	5	
	1	6	5	5	3079
	1	0	0	7	3075
	0	6	2	5	3078
	4	6	0	1	3082

F. Dimensional Warp Drive

target	0	6	0	6	3075
	0	6	5	9	3082
	0	0	0	0	3075
	0	6	2	5	3078
	0	6	0	6	3082

F. Dimensional Warp Drive

target	0	6	0	6	3075
	0	6	5	9	3079
	0	0	0	0	3075
	0	6	2	5	3078
	0	6	0	6	3082

F. Dimensional Warp Drive

target	0	0	9	1	3078
	0	0	3	4	3079
	0	0	0	0	3075
	0	0	0	0	3078
	0	0	9	1	3082

F. Dimensional Warp Drive

target	0	0	9	1	3078
	0	0	3	4	3079
	0	0	0	0	3075
	0	0	0	0	3078
	0	0	9	1	3082

F. Dimensional Warp Drive

target	0	0	0	0	3079
	0	0	0	0	3079
	0	0	0	0	3075
	0	0	0	0	3078
	0	0	0	0	3082

E. The Pharaoh’s Curse

E. The Pharaoh’s Curse

- Breadth first search

E. The Pharaoh's Curse

- Breadth first search
- `seen[me][sarcophagus1][sarcophagus2]`

E. The Pharaoh's Curse

- Breadth first search
- `seen[me][sarcophagus1][sarcophagus2]`
- Needs 100^3 space.

B. Gearbox

B. Gearbox

- Determine rotation speed of each rod

B. Gearbox

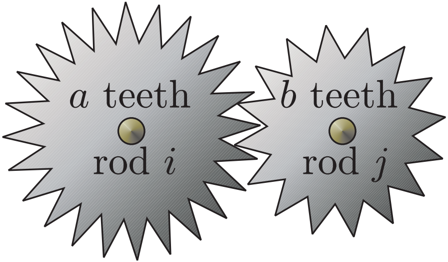
- Determine rotation speed of each rod
- Look for conflicts

B. Gearbox

- Determine rotation speed of each rod
- Look for conflicts
- Start at any rod, set $v_1 = 1$

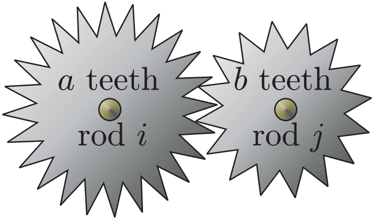
B. Gearbox

- Determine rotation speed of each rod
- Look for conflicts
- Start at any rod, set $v_1 = 1$



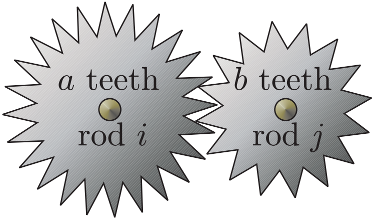
- Interlocking: $v_j = -a/b \cdot v_i$

B. Gearbox



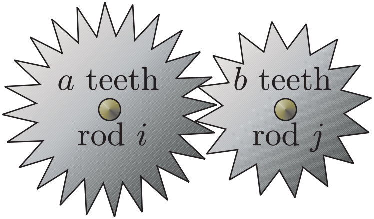
- Interlocking: $v_j = -a/b \cdot v_i$.

B. Gearbox



- Interlocking: $v_j = -a/b \cdot v_i$.
- Symbolically

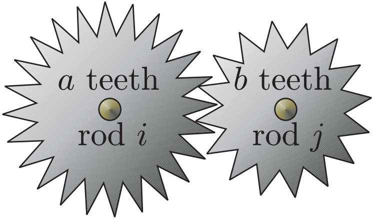
B. Gearbox



- Interlocking: $v_j = -a/b \cdot v_i$.
- Symbolically, for example:

$$v_i = a^3 \cdot b^2 \cdot c^{-5}$$

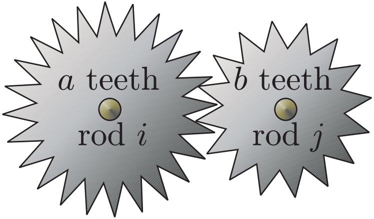
B. Gearbox



- Interlocking: $v_j = -a/b \cdot v_i$.
- Symbolically, for example:

$$v_i = a^3 \cdot b^2 \cdot c^{-5}$$
$$v_j = -a^4 \cdot b^1 \cdot c^{-5}$$

B. Gearbox



- Interlocking: $v_j = -a/b \cdot v_i$.
- Symbolically, for example:

$$v_i = a^3 \cdot b^2 \cdot c^{-5}$$
$$v_j = -a^4 \cdot b^1 \cdot c^{-5}$$

- Store exponents in an array.

H. No Smoking, Please

H. No Smoking, Please

- Minimal Cut

H. No Smoking, Please

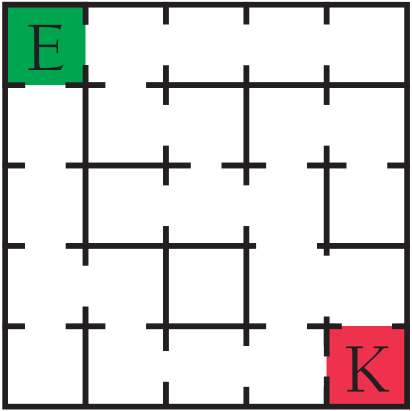
- Minimal Cut $=$ Maximal Flow

H. No Smoking, Please

- Minimal Cut = Maximal Flow
- Walls don't need hatches

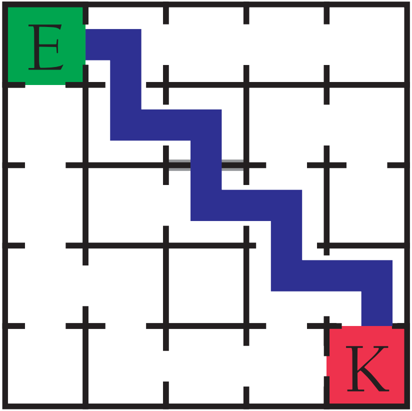
H. No Smoking, Please

- Given a map...



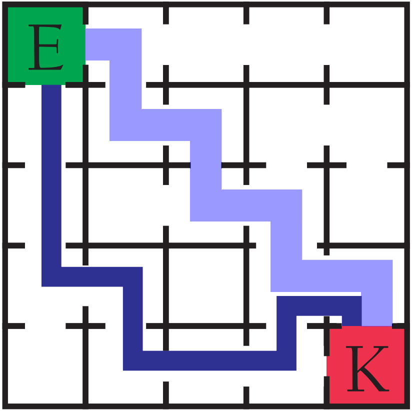
H. No Smoking, Please

- Find an augmenting path from entrance to kitchen



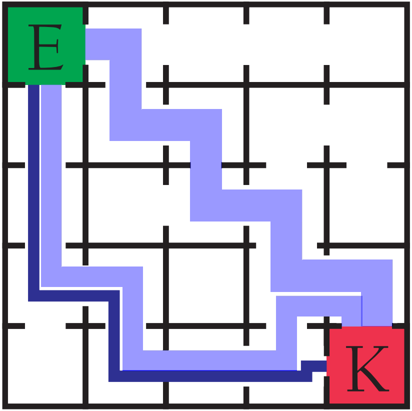
H. No Smoking, Please

- Find an augmenting path from entrance to kitchen



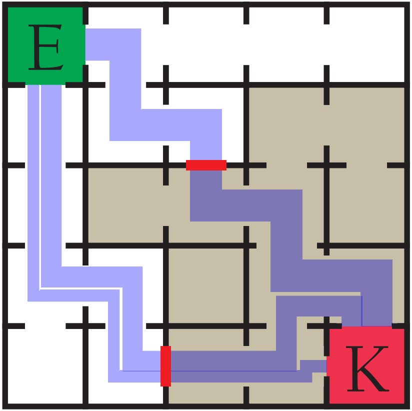
H. No Smoking, Please

- Find an augmenting path from entrance to kitchen



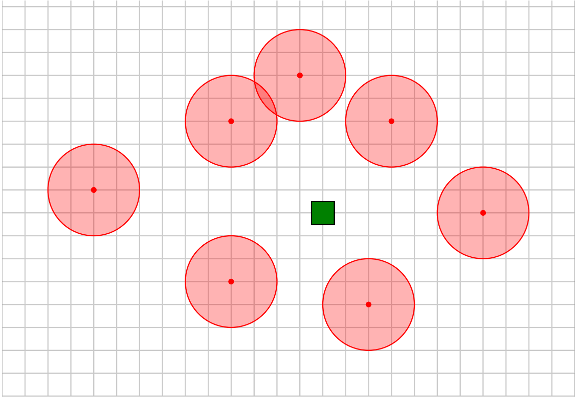
H. No Smoking, Please

- This gives a zoning (which you don't need)



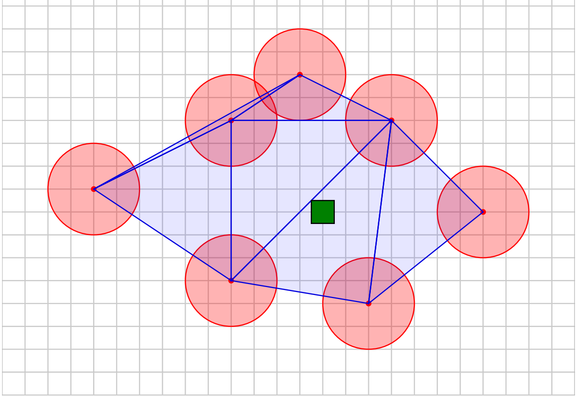
C. Escape from the Minefield

- Given a set of mines



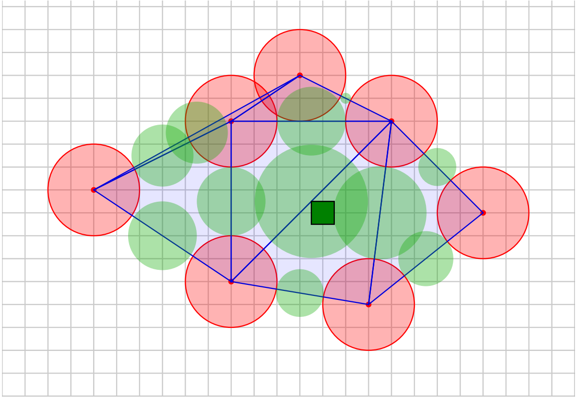
C. Escape from the Minefield

- Construct a Delauney triangulation



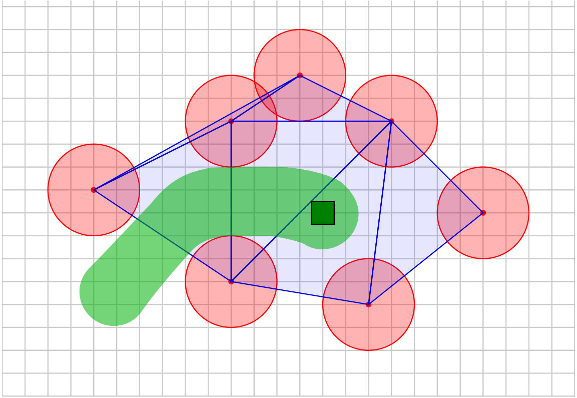
C. Escape from the Minefield

- Find the widest path through the triangulation



C. Escape from the Minefield

- Find the widest path through the triangulation



C. Escape from the Minefield

- Finding the widest path
- Like finding the shortest path, only use min instead of $+$
- Use Dijkstra's algorithm.

Closing Remarks

- Account contents will be emailed.
- Fotos, problem set, test sets will be online soon.

Benelux
Algorithm
Programming
Contest
2009

Dinner Time