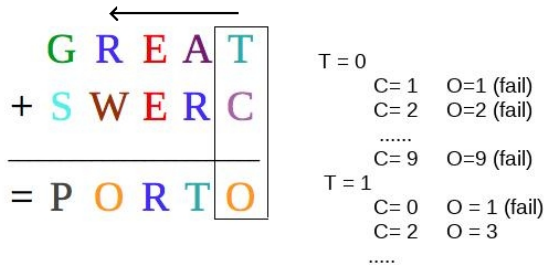
Problem Analysis Session

SWERC Judges

SWERC 2014

A - GREAT + SWERC = PORTO

```
  G R E A T
+ S W E R C
-----------
= P O R T O
```



Problem

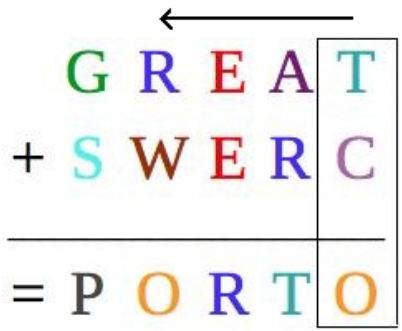
Given a word addition problem, e.g. GREAT + SWERC = PORTO, compute the number of solutions. i.e., how many distinct digit assignments yield a correct sum.

Classification

- Categories: Backtracking (+pruning) or Brute force enumeration
- Difficulty: Easy

A - GREAT + SWERC = PORTO

Sample Solution



T = 0
- C= 1 O=1 (fail)
- C= 2 O=2 (fail)
- .....
- C= 9 O=9 (fail)

T = 1
- C= 0 O = 1 (fail)
- C= 2 O = 3
- .....

- Backtracking (+pruning)
  - The more letters and words, the easier it is to prune an exhaustive search.
  - Process columns from right to left and prune bad solutions earlier. . . (unnecessary optimization).
- Brute force enumeration: generate & test
  - There are at most 10 distinct letters.
    $10! = 3\,628\,800$ candidate solutions. We can just try them all!

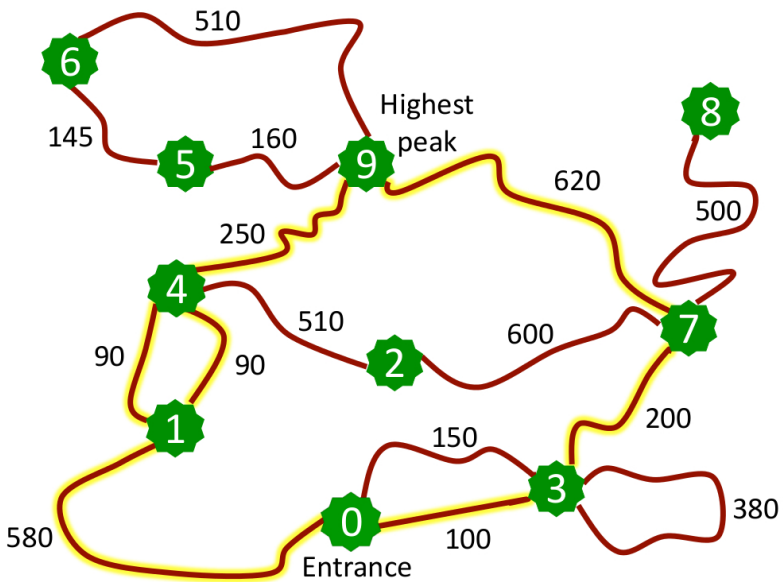
B - Flowery Trails

Problem

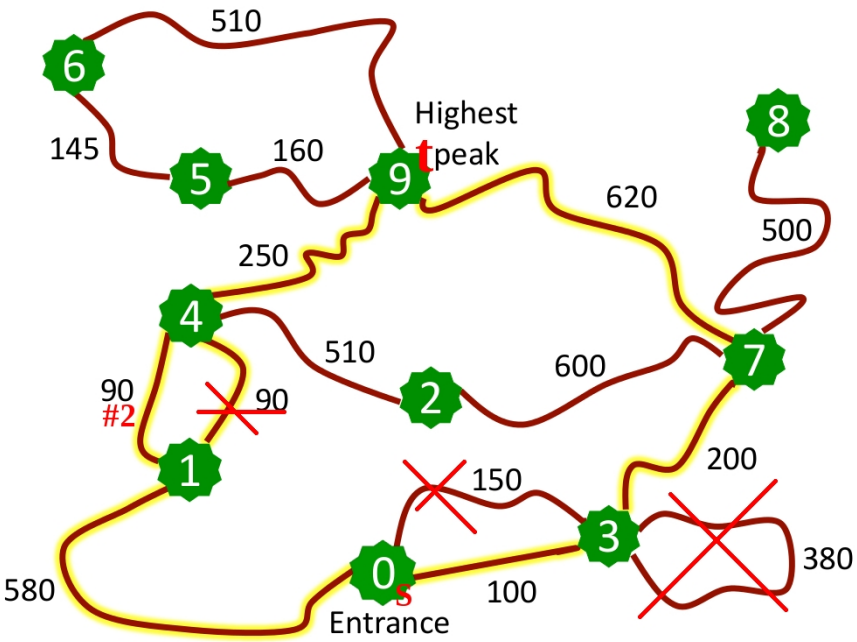
Given an undirected multigraph, determine the edges that are part of a shortest path between vertices 0 (source) and P-1 (target).

Classification

- Categories: Graph Theory, Shortest Paths
- Difficulty: Easy/Medium



B - Flowery Trails



#Flowers = Twice the length of edges that occur in the shortest paths from s to t

Sample Solution ($\mathcal{O}(E \log N)$)

- Adjacency list: keep at most 1 edge $\langle u, v \rangle$, $\#\langle u, v \rangle$, $dist(u, v)$.
- Dijkstra's algorithm to find the shortest path from s to t and the nodes that precede immediately each node in shortest paths.
- BSF/DFS from t to s to "count" flowers. [or use Dijkstra's algorithm twice]

D - Book Club

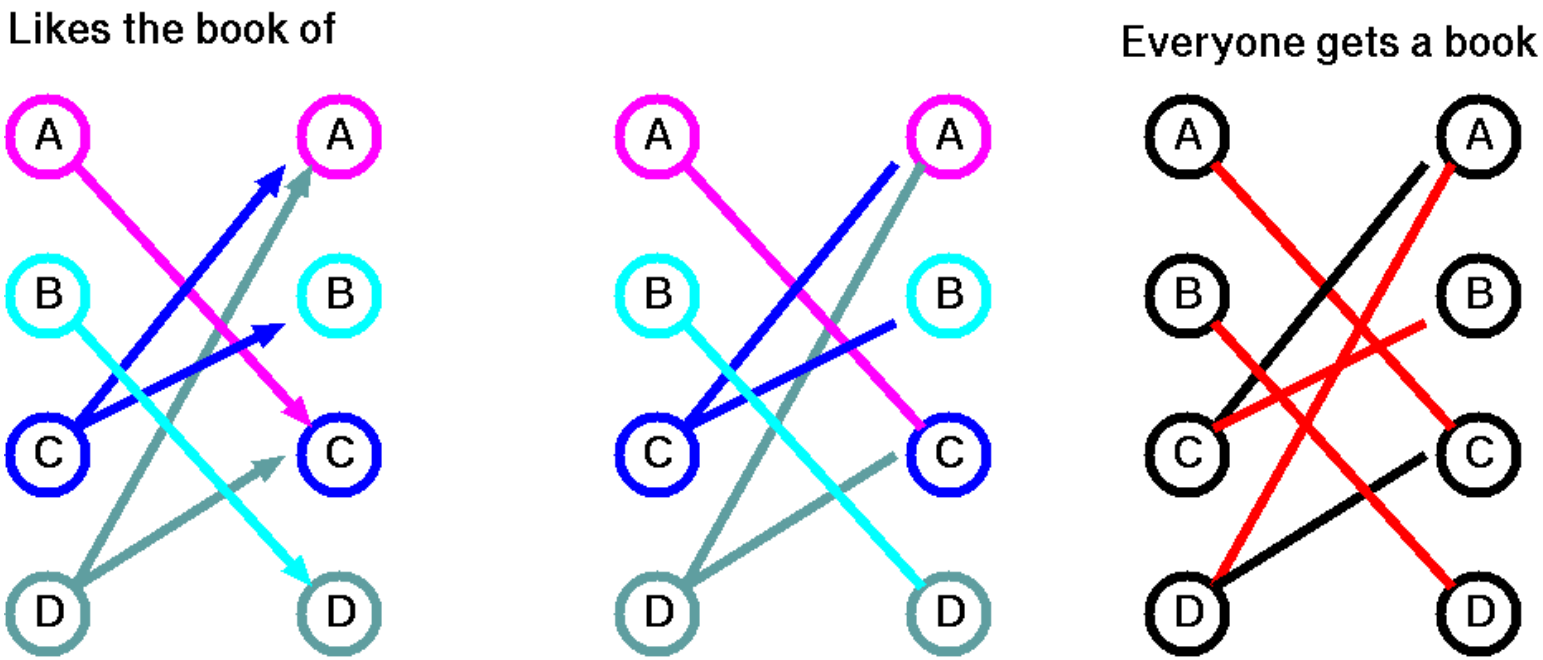
Problem

Given a directed graph where each edge (A, B) means that person A likes the book person B has, is it possible for the group of people to exchange books, so that each one receives a book (s)he likes?

Classification

- Categories: Graph Theory, Maximum Bipartite Matching
- Difficulty: Medium

D - Book Club



Sample Solution

- Bipartite graph: $\langle X, Y \rangle$ if X likes Y's book.
- The answer is *YES* iff there is a perfect matching.
- Either the Hopcroft-Karp algorithm or any maximum flow algorithm was accepted.

E - Ricochet Robots

Problem

Given a matrix representing a floor plant marking free spaces, walls, initial robots positions and target position, is it possible for robot 1 to reach the target position by always moving one of the robots NSWE until hiting an obstacle?

Classification

- Categories: Search
- Difficulty: Easy/Medium

E - Ricochet Robots

Sample Solution

- The floor plant is small, at most 10x10 cells.
- The robots always move until hitting an obstacle. Hence the search space is far smaller than $(10 \times 10)^N$
- Use Breadth First Search where the state is the robots positions
- Save visited states using a hash table or a binary search tree
- Finish if the given depth limit didn't yield a solution.
- Possible (unnecessary) optimization: robots 2 to 4 positions order does not matter. We can reduce the search space by sorting their positions before saving the state

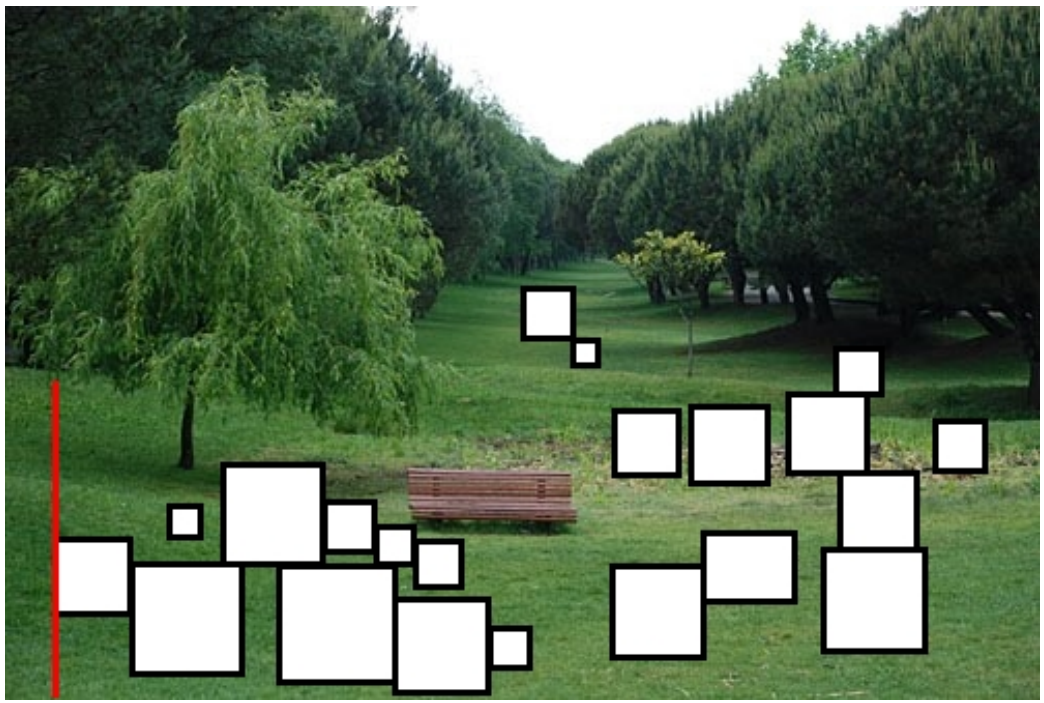
F - City Park

Problem

Given a list of non-overlapping rectangles, what is the area of the largest connected surface of rectangles? Rectangles form a surface if they are tangent.

Classification

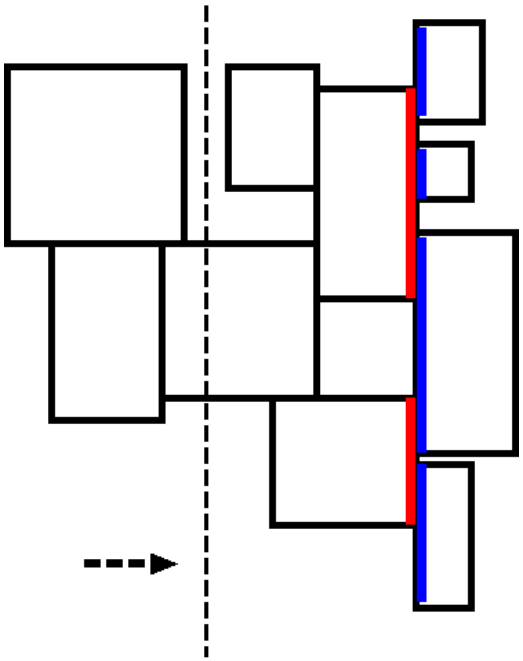
- Categories: Geometry
- Difficulty: Medium



F - City Park

Sample Solution

- No overlaps $\Rightarrow$ #tangencies is $\mathcal{O}(n)$, for n rectangles.
- We must look for a $O(n \log n)$ solution. [$\mathcal{O}(n^2)$ gives TLE].



- How we discover tangencies?
  Exploit a geometric property:
  - Sort closing V-edges and opening V-edges by x-coordinate (and, for each x, sort by y-coordinate). V-tangencies: a kind of "Merge" of the two lists.
    Repeat for H-edges to find H-tangencies.
  - Combine with Union-Find to obtain the connected components or build the adjacency graph and apply BFS.
- Or adapt plane-sweep: a sweep V-line from left to right; suport sweepline status of active rectangles by a binary search tree (BST).

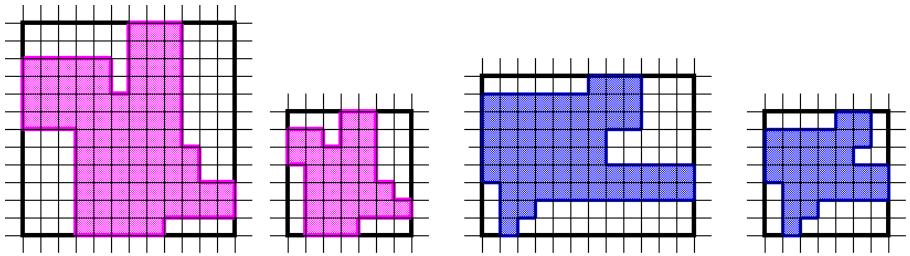
G - Playing with Geometry

Problem

Given two rectilinear polygons with no collinear edges, is it possible to transform them to the same permutomino using only slides and turns?

Classification

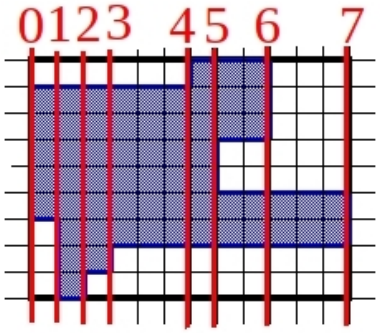
- Categories: Geometry
- Difficulty: Easy

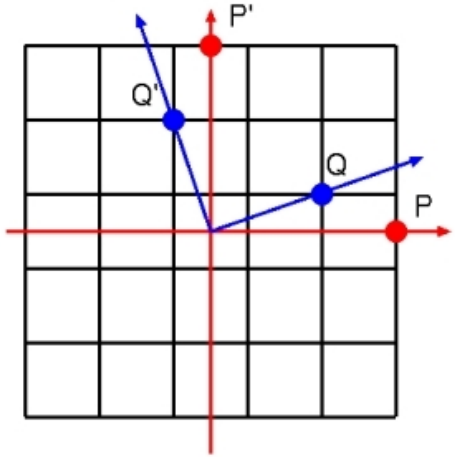


G - Playing with Geometry

Sample Solution $\mathcal{O}(n \log n)$ (if we "sort" edges to remove the empty grid lines)
Her teacher has explained how slides and turns act on point coordinates, but, by that time, Alice was already thinking about an episode of "The Sympsons"...

- Slides to remove empty lines: change vertex coordinates. $0 \leq x, y \leq 3000$ too small; not necessary to sort edges; use 0/1 array instead.
- Rotation by 90° CCW: $(X, Y) \rightarrow \left((\frac{n}{2} - 1) - Y, X\right)$; by 180° CCW: $(X, Y) \rightarrow \left(-X + (\frac{n}{2} - 1)), -Y + (\frac{n}{2} - 1)\right)$; ... Fix one polygon and rotate the other... (optimization: check the minimum bounding box first).





The Safe Secret

Problem

Given a sequence of k numbers and k operators (sum, subtraction, multiplication or any of the previous three), find, for every rotation of the sequence, the minimum and the maximum values we can obtain by evaluating the operations in any order

Classification

- Categories: Dynamic Programming
- Difficulty: Medium

I - The Safe Secret

Base Approach

- Let's first treat the problem without circularity.
  It's similar to the matrix chain multiplication problem.
- The "max rules":

$$
\begin{aligned}
\max(X + Y) &= \max(X) + \max(Y) \\
\max(X - Y) &= \max(X) - \min(Y) \\
\max(X * Y) &= \max(\ \min(X) * \min(Y),\quad \min(X) * \max(Y), \\
&\qquad\qquad \max(X) * \min(Y),\quad \max(X) * \max(Y)\ ) \\
\max(X\ ?\ Y) &= \max(\ \max(X + Y),\ \max(X - Y),\ \max(X * Y)\ )
\end{aligned}
$$

- Time complexity of the dynamic programming algorithm:
  $\mathcal{O}(k^3)$, for 1 expression;
  $\mathcal{O}(k^4)$, for k expressions, treated independently (which is too high).

Optimizing the base approach

- The k expressions share many sub-problems (example for $k = 4$)

1 expression :

$\frac{k(k+1)}{2}$

sub-problems

n_0	n_1	n_2	n_3
$n_0 n_1$	$n_1 n_2$	$n_2 n_3$	
$n_0 n_1 n_2$	$n_1 n_2 n_3$		
$n_0 n_1 n_2 n_3$			

k expressions :

k^2

sub-problems

n_0	n_1	n_2	n_3
$n_0 n_1$	$n_1 n_2$	$n_2 n_3$	$n_3 n_0$
$n_0 n_1 n_2$	$n_1 n_2 n_3$	$n_2 n_3 n_0$	$n_3 n_0 n_1$
$n_0 n_1 n_2 n_3$	$n_1 n_2 n_3 n_0$	$n_2 n_3 n_0 n_1$	$n_3 n_0 n_1 n_2$

- Time complexity of the dynamic programming algorithm: $\mathcal{O}(k^3)$, for k expressions.

J - The Big Painting

Problem

Given a binary matrix and a 2d pattern, how many times does the pattern occur in the matrix?

Classification

- Categories: String Matching
- Difficulty: Medium-Hard

J - The Big Painting

Sample Solution

- With $N = L = C$, a brute force approach takes $\mathcal{O}(N^4)$
- Treat equal pattern lines as the same
- The sample input line sequence would be $1, 2, 2, 1$ because the 1^{st} line equals the 4^{th} and the 2^{nd} equals the 3^{rd}.
- For each line, find the positions where each pattern line occurs

- The pattern occurs whenever there is a sequence $1, 2, 2, 1$ in the columns of the matrix on the right

J - The Big Painting

Sample Solution

- Aho-Corasick algorithm to find the positions where each pattern line occurs in each line of text - $\mathcal{O}(N)$ - taking $\mathcal{O}(N^2)$ total
- Search the pattern line sequence for each of the N columns in $\mathcal{O}(N)$ time using Aho-Corasick or Knuth-Morris-Pratt algorithms.
  This also takes $\mathcal{O}(N^2)$ total time
- The overall time complexity of this approach is $\mathcal{O}(N^2)$

- Extending the Rabin-Karp algorithm to 2 dimensions would also be a valid approach

C - Golf Bot

Problem

Given two sets of integers, Distances and Holes, how many integers from set Holes can be formed by summing up to two (possibly the same) integers from Distances set?

Classification

- Categories: Math, Fast Fourier Transform
- Difficulty: Hard

- The problem seems easy until we check the constraints.
- A simple approach is to hash the possible distances. Then we could check each hole in $\mathcal{O}(N)$ time.
- However, this would yield an $\mathcal{O}(N^2)$ algorithm which is too slow.

C - Golf Bot

Sample Solution

- Let's restate the problem and use binary vectors.
  - *S[i]* is 1 if golf bot can shoot distance *i*
  - *H[i]* is 1 if there is a hole at distance *i*
- We want to calculate $Possible[i] = \sum_{k=0}^{i} S[k] * S[i-k]$
- The *Possible* vector is actually the discrete convolution of S with itself. We can calculate it the same way we multiply polynomials using a discrete Fast Fourier Transform (FFT).

C - Golf Bot

Sample Solution

- The discrete FFT of n values can be calculated in $\mathcal{O}(n \log n)$
- After calculating the *Possible* vector, we just need to check in how many distances i, $Possible[i] > 0$ and $H[i] = 1$
- Recall that the distances are up to 200 000. Say the maximum distance is D, then the time complexity of this approach is $\mathcal{O}(N + M + D \log D)$

H - Money Transfers

Problem

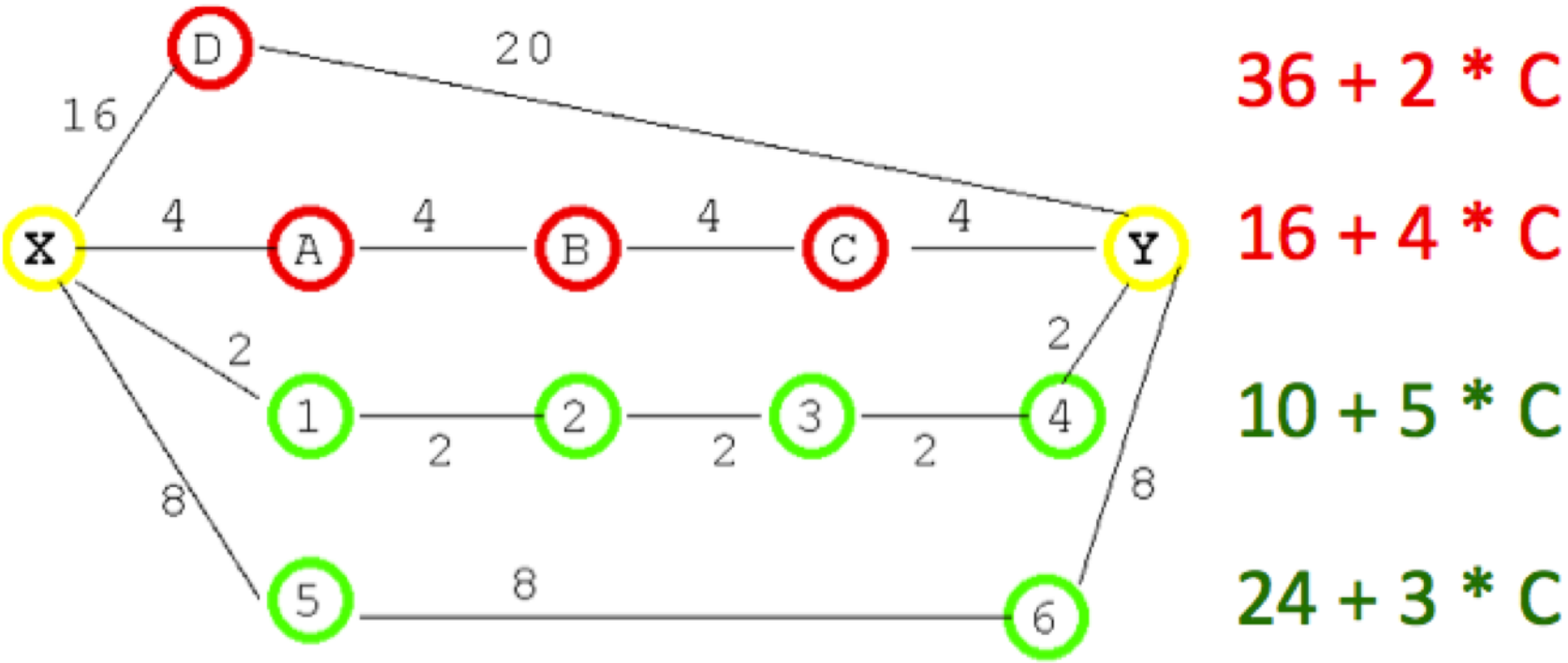
Given a weighted undirected graph and a subset M of its vertices, what is the maximum weight we can add to each edge of the graph so that the shortest path between X and Y passes only through vertices of M?

Classification

- Categories: Graph Theory, Shortest Paths
- Difficulty: Hard

H - Money Transfers

Sample Solution



- For any 2 paths with i and j edges and $i < j$, it's always possible to add a large enough cost so that $w_i + i * C < w_j + j * C$

H - Money Transfers

$$\max C: \quad \exists s \in \mathcal{D}istOurs \;\; \forall r \in DistTheirs \quad w_s + n_s C < w_r + n_r C$$

Sample Solution

- Find the shortest path from X to Y, using i edges for $1 \leq i < N$, both using vertices in M and other vertices.
- Let A be the minimum number of edges in a path using only vertices from M and $DistOurs[A]$ the minimum cost with A edges
- Similarly, let B be the minimum number of edges in a path using other vertices and $DistTheirs[B]$ the minimum cost with B edges
- The answer is *Infinity* if
  $A < B$ or $(A = B$ and $DistOurs[A] < DistTheirs[B])$

H - Money Transfers

Sample Solution

- For each shortest path with i edges using vertices in M, compare it with the paths that use other vertices

$$\max C: \quad \exists s \in \mathcal{D}istOurs \;\; \forall r \in DistTheirs \quad w_s + n_s C < w_r + n_r C$$

- Bellman-Ford in $\mathcal{O}(P * N)$ is simple and fast enough for the first step
- We can just compare all paths with i and j edges in $\mathcal{O}(N^2)$
- The time complexity of this approach is $\mathcal{O}(P * N + N^2)$

- Using Dijkstra's algorithm for the first step would lead to a more efficient algorithm. The state is a composite key $(cost, num\_edges, only\_ours?)$ and we can ignore irrelevant paths