BAPC 2019 Preliminaries

Solutions presentation

September 22, 2019

Architecture

- Does there exist a city with the two given skylines?
- Let the tallest building have height h.
- The maximal height in the eastern skyline is h.
- The maximal height in the northern skyline is h.
- A necessary condition is that $\max x_i = h = \max y_j$.
- It is also sufficient:
  Find r and c with $x_r = y_c = h$ and set $h_{rj} = y_j$ and $h_{ic} = x_i$.

0	0	3	0
4	1	6	3
0	0	1	0
0	0	2	0

Architecture

```
input()
if max([int(x) for x in input().split()])
        == max([int(x) for x in input().split()]):
    print("possible")
else:
    print("impossible")
```

Bracket Sequence

- Build an expression tree and evaluate it.
- Be careful to put the $+$ and $\times$ at the right levels!
- Implement using recursion, a stack, or linked lists.
- Instead of computing levels 'outside-in', you can also compute the value of each subexpression for both the $+$ and $\times$ case and decide which one you need at the end.

- Python `eval` goes a long way, but stackoverflows.

Canyon Crossing

- What is the lowest height where we can make a path using at most k bridges?
- If we can do it with minimal height h, we can also do it for all $h' \geq h$.
- Binary search for h.
- For each h, we can do a BFS where for each cell we store the number of bridges needed to get there.
- If we can reach the other side with at most k bridges: answer $\leq h$. Else: answer $> h$.
- Dijkstra instead of BFS will be too slow.

Deceptive Dice

- Given: a die with n sides, k rolls.
- Using our best strategy, what is our expected score?
- Example: given $n = 20$ sides and $k = 1$ roll, our expected score is

$$\frac{1 + 2 + \cdots + 19 + 20}{20} = 10\frac{1}{2}.$$

If we have $k = 2$ rolls, we want to reroll if our first result $< 10\frac{1}{2}$. So our expected score is

$$\frac{11 + 12 + \cdots + 20}{20} + \frac{10 \times 10\frac{1}{2}}{20} = 13.$$

So for $k = 3$ rolls, we reroll if our first result < 13. Score for 3 rolls:

$$\frac{14 + \cdots + 20}{20} + 13 \times \frac{13}{20} = 14\frac{2}{5}.$$

And so on, until we reach k rolls.

- A linear solution is possible by computing the sums in constant time.

Exits in Excess

- Given a directed graph, remove at most half the edges so that it becomes acyclic. Lots of ways to do this. Here is one way:
- Partition the edges into two sets U and D such that both are acyclic.
- For each edge $u \to v$:
  - If $u < v$, put it in U.
  - If $u > v$, put it in D.
- If U is smaller, output all edges in U. Otherwise, output all edges in D.
- There cannot be cycles in U: along every edge the number of the node goes up. And vice versa for D.

Floor Plan

- Given $1 \leq n \leq 10^9$, find two integers m and k solving
$$n = m^2 - k^2.$$
- Linear solution: Try all m between $\sqrt{n}$ and $2n$. Takes $> 10^9$ steps, so too slow!
- Let's try some simple examples:
$$(m+1)^2 - m^2 = 2m + 1.$$
So we can make all odd numbers this way.
$$(m+2)^2 - m^2 = 4m + 4.$$
So we can make all multiples of 4 this way.
- What about if n is even but not divisible by 2?
$$n = m^2 - k^2 = (m-k)(m+k).$$
If n is even, then at least one of $m-k, m+k$ is even. But then they are both even, so $4 \mid n$. Conclusion: `impossible`.

Greetings!

- Read the input and print the output with twice the number of e's.
-

```
s = input()
print(s[0] + s[1:-1] + s[1:-1] + s[-1])

print('h' + 'e'*(len(input())*2-4) + 'y')

print(input().replace('e','ee'))
```

Greetings!

```
hey = input()
print("he" + hey[2:-2] * 2 + "ey")
```

Greetings!

```
hey = input()
print("h" + hey[1:-1] * 2 + "y")
```

Greetings!

Why not try something quadratic?

```
int main(){
    char s[2001];
    cin.get(s, 1001);
    for(int i=1; i < strlen(s); ++i){
        if(strchr("e", s[i])){
            for(int j = strlen(s)+1; j > i; --j){
                s[j] = s[j-1];
            }
            ++i;
        }
    }
    cout << s << '\n';
    return 0;
}
```

Hexagonal Rooks

- Given a hexagonal chess board with a rook on it, in how many ways can the rook move to a target cell in exactly two steps?
- For each cell on the board:
  - Check that you can go from the start to this cell, and to the goal from this cell.
  - Check that the cell is not equal to the start or the goal.

Inquiry I

- What is the maximal value of

$$(a_1^2 + \cdots + a_k^2) \cdot (a_{k+1} + \cdots + a_n)?$$

- Trying all $n-1$ possible values of k separately takes $O(n^2)$ time: Too slow!
- We can do it in linear time by remembering the partial sums of $\sum_i a_i^2$ and $\sum_i a_i$:

```
n = int(input())
a = [int(input()) for _ in range(n)]
l, r = 0, sum(a)
best = 0
for x in a:
    l += x*x
    r -= x
    best = max(best, l*r)
print(best)
```

Jumbled Journey

- Given a table of average distances between vertices, reconstruct the original directed graph.
- To compute the length of edge $u \to v$ and whether it's present, we must first know all other edges on the path from u to v.
- Toposort the vertices, and start by processing all adjacent vertices. Then process vertices at longer distances.
- Keep track of three tables: the input `avg_dist[u][v]`, the number of paths `count[u][v]`, and the length of the edge, if present `edge[u][v]`.
- The number of paths c from u to v and their total length L can be calculated by looping over the last vertex w of the path before v.
- If the average distance is not already correct add the edge $u \to v$ with length l such that

$$(l + L)/(c + 1) = \text{avg}_{u,v}.$$

Knapsack Packing

- Given a set of 2^n integers S find a integers $a_1, \ldots, a_n$ such that the set of the sums of all subsets is S:

$$\left\{ \sum_{i \in I} a_i \middle| I \subseteq \{1, 2, \ldots, n\} \right\} = S.$$

- $0 \in S$ because it's the sum of the empty set.
- $\min_i a_i \in S$ and must the the next smallest element.
- Add this value m to the solution and for each value x (in increasing order) remove $x + m$ from S.
- Repeat until S contains only 0.
- Be careful to print `impossible` when needed!

Knapsack Packing

- Given a set of 2^n integers S find a integers $a_1, \dots, a_n$ such that the set of the sums of all subsets is S:

$$\left\{ \sum_{i \in I} a_i \,\middle|\, I \subseteq \{1, 2, \dots, n\} \right\} = S.$$

$$\{0, 1, 3, 3, 4, 4, 6, 7\}$$

$$\{0, 1, 3, 3, 4, 4, 6, 7\}$$

$$\{0, 1, 3, 3, \cancel{4}, 4, 6, 7\}$$

$$\{0, 1, 3, 3, \cancel{4}, \cancel{4}, 6, 7\}$$

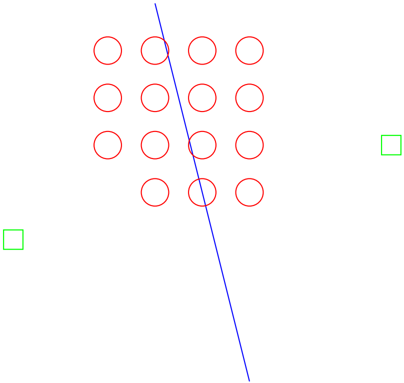
$$\{0, 1, 3, 3, \cancel{4}, \cancel{4}, 6, \cancel{7}\}$$

Lifeguards

- Given a set of points, find a line that evenly devides them into two equally sized groups.
- In the odd case, the line must go through exactly one point.
- Idea: Find the *middle point* and move/rotate the line slightly.
- Sort by (x, y) and take the middle point.
- For large M, the line through $(x - M, y - 1)$ and $(x + M, y + 1)$ goes through (x, y) and no other points.
- In the even case use $(x - M, y - 1)$ and $(x + M, y + 0)$ instead.

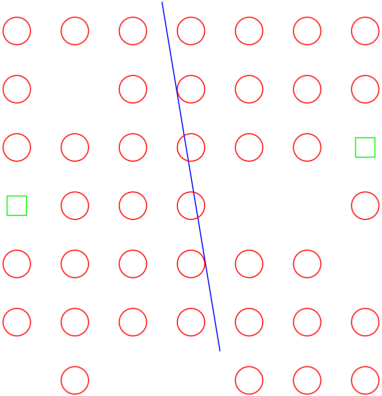
Lifeguards

Odd: go through the middle point.



Lifeguards

Even: Go just under the 'middle' point.



Some stats

- 400 commits
- 480 testcases
- 170 jury solutions
- Each problem but Canyon Crossing can be solved with Python!
- The number of lines needed to solve all problems is

$$2 + 7 + 39 + 4 + 9 + 4 + 1 + 20 + 7 + 25 + 16 + 13 = 147.$$

On average 12.3 lines per problem!

The Jury

- Ragnar Groot Koerkamp
- Mees de Vries
- David Venhoek
- Harry Smit
- Daan van Gent
- Wessel van Woerden
- Timon Knigge
- Bjarki Ágúst Guðmundsson
- Onno Berrevoets