Christmas Presents
Threefold Problem Set

ETH Zürich

December 14, 2016

Fetch two sheets of paper at the entrance!

Threefold Problem Set

Goal: simulate the thinking steps of a six hour exam in one hour.

First Hour: (17:15 – 18:00)

- 3 problems on paper
- think about them
- sketch your solution on paper
- no coding required
- provide us with some feedback

Second Hour: (18:15 – 19:00)

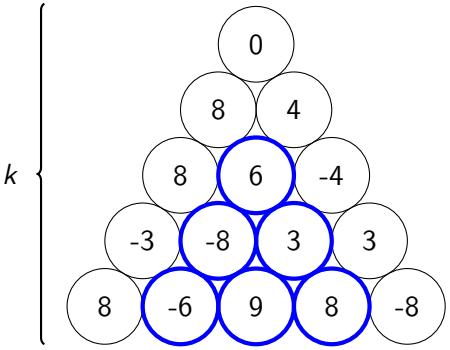
- solution discussion

Afterwards: (19:00 –)

- Q &A session

Fetch two sheets of paper at the entrance!

Alice's Accumulation – naive solution



Goal: maximize the sum of a sub-pile.

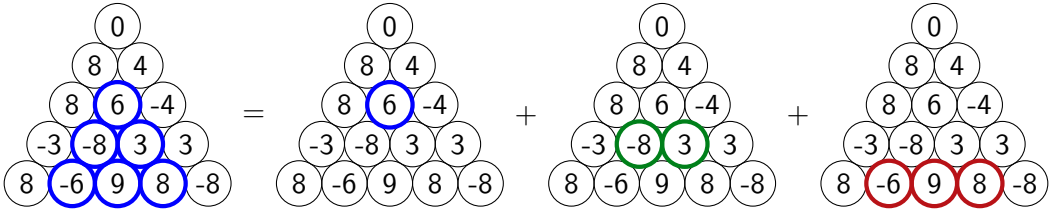
Naive attempt: for each package sum up the values in the pile below that package.

Running time: $\Omega(k^4)$.

(There are $\sim k^2/4$ packages in the top half and each of them has $\sim k^2/4$ packages below them.)

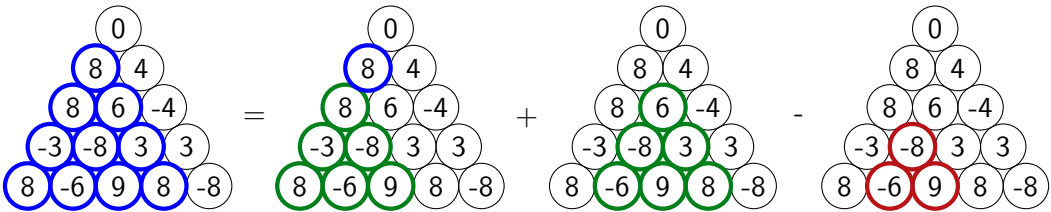
Alice's Accumulation – partial sums

Idea: precompute partial sums in each row (in time $O(k^2)$). Then we get a solution with running time $O(k^3)$:



Alice's Accumulation – recursion

Idea: find a recursion.



Alice's Accumulation – dynamic program

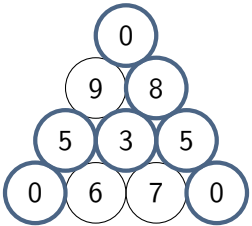
Hence, dynamic programming. Running time: $O(k^2)$.

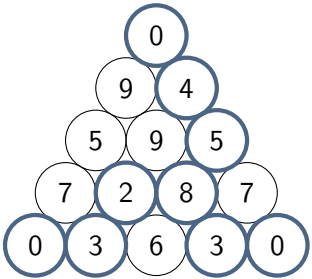
```
1   // Given: D[i][j] = value of the j-th ball in the i-th row
2   // Compute: P[i][j] = sum of sub-pile rooted at (i,j)
3
4   for (int i = k-1; i >= 0; --i) {
5     for (int j = 0; j < i; ++j) {
6       if (i == k-1)
7         P[i][j] = D[i][j];
8       else if (i == k-2)
9         P[i][j] = D[i][j] + P[i+1][j] + P[i+1][j+1];
10      else
11        P[i][j] = D[i][j] + P[i+1][j] + P[i+1][j+1] - P[i+2][j+1];
12    }
13  }
```

Bob's Burden – understanding the problem

Task: Find center ball
with minimum
weight + distances.

Weight: own weight of ball
Distances: to triangle apex
Center: may not be unique





Modeling the distance:

- B_{in} and B_{out} for each ball B,
- interior edge with the ball's weight,
- incoming/outgoing 0-edges to neighbors.

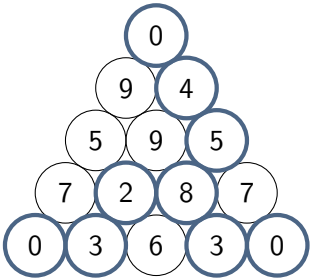
Bob's Burden – small subtask

First subtask: $k \leq 40 \Rightarrow \approx 800$ balls

For each candidate ball B,
we can compute the distances to each triangle apex:

Either by running Dijkstra from B_{out}.
In this case we read the distances stored at $B11_{\text{out}}, Bk1_{\text{out}}, Bkk_{\text{out}}$.

Or by running a MinCost MaxFlow (SSP version) from B_{out} to a sink reachable from $B11_{\text{out}}, Bk1_{\text{out}}, Bkk_{\text{out}}$.
In this case we get the sum of the distances with `find_flow_cost(G)`.



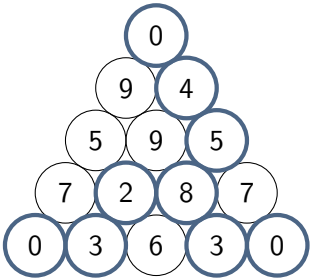
Note: for the correct central ball (weight 8) we get the correct sum to the apices, for any other ball (e.g. the ball of weight 2) an overestimate, which is fine.

Bob's Burden – full solution

Full solution: $k \leq 800 \Rightarrow \approx 320'000$ balls

We probably still have to look at each ball as a candidate.
How can we avoid the many (costly) Dijkstra runs?

Do Dijkstra **only 3 times**, from $B11_{\text{out}}$, $Bk1_{\text{out}}$ and Bkk_{out}!
Details: Use 3 exterior property maps `distmap`i, $i = 1, 2, 3$.
Access distances with `distmap`i`[`B_{in}`]`.
Add the weight of the ball B.



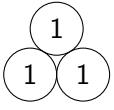
Remark: Simple testdata available on the judge.

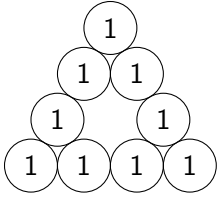
Carol's Configuration – boundary only

This case is quite easy to solve "by hand".

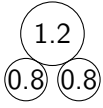
Note that the balls on the boundary form a cyclic structure.
First, assume that k is even: Then the cycle has odd length.

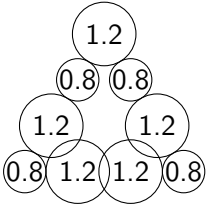
The cycle stays connected
if and only if all radii are 1.





If any radius is different than 1,
then it will be disconnected or intersecting.



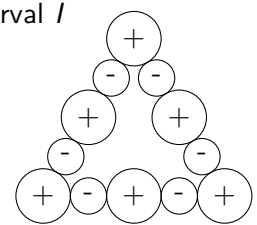


Carol's Configuration – boundary only

Now assume that k is odd. Then the cycle has even length.

- Set the radius of B_{11} at $1 + x$, $0 \leq x \leq 1$. Note that it cannot be smaller than 1 as then its two neighbors will have radius >1 and they will intersect.
- The situation is identical for all three "corners" of the triangle.
- The radii on the cycle alternate between $1 + x$ and $1 - x$ such that the boundary remains connected.

The result is a linear function of the form $(1 + x)c_1 + (1 - x)c_2 = c3 + c4 \cdot x$.
Any monotone function takes its maximum value in a closed interval I at the boundary of I, i.e. at $x = 0$ or $x = 1$.



Carol's Configuration – full solution

For the general case we will formulate a Linear Program.

$$
\begin{array}{lll}
\text{maximize} & \sum r_{ij} v_{ij} & \\
\text{subject to} & r_{ij} + r_{i'j'} \leq 2, & \text{for all } 1 < j < i < k \text{ and } i'j' \in N_1(ij) \\
& r_{ij} + r_{i'j'} \leq 2\sqrt{3}, & \text{for all } 1 < j < i < k \text{ and } i'j' \in N_2(ij) \\
& r_{i1} + r_{(i+1)1} = 2 & \text{for all } i < k \\
& r_{ii} + r_{(i+1)(i+1)} = 2 & \text{for all } i < k \\
& r_{kj} + r_{k(j+1)} = 2 & \text{for all } j < k
\end{array}
$$

Where $N_1(ij)$ is the set of direct neighbors of B_{ij} and $N_2(ij)$ is the set of indirect neighbors. That is, all the balls that might intersect with B_{ij}.
For all ij we have $|N_1(ij)| \leq 6$ and $|N_2(ij)| \leq 6$.