Wonderful City

Calvin Fung

July 1, 2025

1 Problem

You are the proud leader of a city in Ancient Berland. There are n^2 buildings arranged in a grid of n rows and n columns. The height of the building in row i and column j is $h_{i,j}$.

The city is *beautiful* if no two adjacent by side buildings have the same height. In other words, it must satisfy the following:

- There **does not** exist a position (i, j) ($1 \le i \le n$, $1 \le j \le n-1$) such that $h_{i,j} = h_{i,j+1}$.
- There **does not** exist a position (i, j) ($1 \le i \le n-1$, $1 \le j \le n$) such that $h_{i,j} = h_{i+1,j}$.

There are n workers at company A, and n workers at company B. Each worker can be hired **at most once**.

It costs a_i coins to hire worker i at company A. After hiring, worker i will:

- Increase the heights of all buildings in row i by 1. In other words, increase $h_{i,1}, h_{i,2}, \ldots, h_{i,n}$ by 1.

It costs b_j coins to hire worker j at company B. After hiring, worker j will:

- Increase the heights of all buildings in column j by 1. In other words, increase $h_{1,j}, h_{2,j}, \ldots, h_{n,j}$ by 1.

Find the minimum number of coins needed to make the city beautiful, or report that it is impossible.

2 Solution

We present a linear time solution.

2.1 Analysis

We define some terminologies :

- We say the city is *horizontally beautiful* if there doesn't exist a position $(i, j), i \in [1, n], j \in [1, n-1]$ such that $h_{i,j} = h_{i,j+1}$.

- We say the city is *vertically beautiful* if there doesn't exist a position $(i, j), i \in [1, n-1], j \in [1, n]$ such that $h_{i,j} = h_{i+1,j}$.

Let $OPT_G, OPT_{G,H}, OPT_{G,V}$ be the optimal sequence of hiring/operation (such that each worker is hired at most once) to make city G *beautiful*, *horizontally beautiful* and *vertically beautiful* respectively. For any sequence S of hiring let $c(S)$ be the cost.

Also for city G, we denote the hight of the building at position (i, j) as $G[i, j]$.

Lemma 1. Suppose we increase a row i of a city G by 1 to obtain city G'. If either one of $OPT_{G,H}$ or $OPT_{G',H}$ exist, both must exist and we have $c(OPT_{G,H}) = c(OPT_{G',H})$.

proof. If $OPT_{G,H}$ ($OPT_{G',H}$) exist, the first (second) part of proof below implies $OPT_{G',H}$ ($OPT_{G,H}$) exists.

($c(OPT_{G,H}) \ge c(OPT_{G',H})$) Suppose apply $OPT_{G,H}$ on G gives us a *horizontally beautiful* city X. Applying $OPT_{G,H}$ on G' gives a *horizontally beautiful* city Y , otherwise there's some $Y[u, v] = Y[u, v+1]$, then subtracting row i of Y gives us Z and we still have $Z[u,v] = Z[u,v+1]$ so Z is not *horizontally beautiful*. However, it's easy to see that Z must be X, thus a contradiction.

$(c(OPT_{G,H}) \leq c(OPT_{G',H}))$ similar ■

Lemma 2. OPT_G exists iff both $OPT_{G,H}$ and $OPT_{G,V}$ exist.

proof.

($\Rightarrow$) It's obvious that if OPT_G exists, then both $OPT_{G,H}$ and $OPT_{G,V}$ must exist because OPT_G makes G *horizontally beautiful* and *vertically beautiful* too.

($\Leftarrow$) First apply $OPT_{G,V}$ (notice it consists of only hirings from company A) on G to get a *vertically beautiful* city G'. By lemma 1, since $OPT_{G,H}$ exists $OPT_{G',H}$ also exists. So we then apply $OPT_{G',H}$ (notice it consists of only hirings from company B) on G' to get a *horizontally beautiful* city G''. As column operations cannot turn a *vertically beautiful* city into one that's not, G'' has to be *beautiful*. ■

Lemma 3. If OPT_G exists then $c(OPT_G) = c(OPT_{G,H}) + c(OPT_{G,V})$.

proof.

We can assume in OPT_G no worker from company B is hired before a worker from company A because it's easy to see order doesn't matter (to see this focus on 1 entry of G and how this value changes). Therefore we can write $OPT_G = OPT_{G,A}OPT_{G,B}$ where $OPT_{G,A}$ is a subsequence of hiring consisting of only workers from company A (similarly $OPT_{G,B}$ only from company B). Clearly $c(OPT_G) = c(OPT_{G,A}) + c(OPT_{G,B})$. Suppose applying OPT_G on G gives us a *beautiful* city X.

$(c(OPT_{G,A}) \geq c(OPT_{G,V}))$ Suppose applying $OPT_{G,A}$ on G gives us Y. If Y is not *vertically beautiful* then X will not be *vertically beautiful* because column operations $OPT_{G,B}$ cannot make Y *vertically beautiful*.

$(c(OPT_{G,B}) \geq c(OPT_{G,H}))$ By lemma 1, $c(OPT_{G,H}) = c(OPT_{Y,H})$. By definition applying $OPT_{G,B}$ on Y produces *horizontally beautiful* city X.

Thus we have shown $c(OPT_G) \geq c(OPT_{G,V}) + c(OPT_{G,H})$.

To see the other way $c(OPT_G) \leq c(OPT_{G,V}) + c(OPT_{G,H})$, by lemma 2 $OPT_{G,V}$ exists (notice it consists of only hirings from company A) and we can apply it on G to get a *vertically beautiful* city G'. By lemma 2 $OPT_{G,H}$ exists, along with lemma 1 we know $OPT_{G',H}$ exists (notice it consists of only hirings from company B) so we apply it on G' to get a *horizontally beautiful* city G''. As column operations cannot turn a *vertically beautiful* city into one that's not, G'' has to be *beautiful*. Moreover, by lemma 1 $c(OPT_{G,H}) = c(OPT_{G',H})$. ■

2.2 Algorithm

We'll first try to compute $OPT_{G,H}$ and $OPT_{G,V}$, if both exist then return $c(OPT_{G,H}) + c(OPT_{G,V})$, otherwise OPT_G doesn't exist. By lemma 2 and 3 this approach is correct.

Below is a linear time procedure to compute $OPT_{G,V}$, it returns $c(OPT_{G,V})$ if it exists and -1 otherwise.

```
f(G,n,A,B){
        /*
        dp[k,0] is the minimum number of coins of hiring workers from A[1,...,k]
        (such that each is hired at most once) so that there doesn't exist a position
        (i,j) (1 <= i <= k - 1, 1 <= j <= n) such that G[i,j] = G[i+1,j]
        GIVEN THAT A[k] IS NOT HIRED

        If this's infeasible, we set dp[k,0] = inf

        Similarly dp[k,1] for the case where A[k] is hired
        */
        dp = empty n*2 matrix
```

```
        dp[1,0] = 0, dp[1,1] = A[1]
        for(i = 2; i <= n; ++i){
                        p = q = inf
                        if(feasible(i,0,0)) p = dp[i-1,0];
                        if(feasible(i,0,1)) q = dp[i-1,1];
                        dp[i,0] = min(p,q);

                        p = q = inf
                        if(feasible(i,1,0)) p = dp[i-1,0];
                        if(feasible(i,1,1)) q = dp[i-1,1];
                        dp[i,1] = min(p,q) + A[i];
        }
        res = min(dp[n,0],dp[n,1]);
        return res == inf ? -1 : res;
}

/*
It checks after raising row i by x and row i-1 by y to get G', whether there's a 1 <= v <= n
such that G'[i,v] = G'[i-1,v]
*/
feasible(i,x,y){
        for(v = 1; v <= n; ++v){
                if(G[i,v] + x == G[i-1,v] + y)  return false;
        }
        return true;
}
```

The procedure to compute $OPT_{G,H}$ will be similar, it'll also be linear time. Combining those two procedures as described at the beginning gives us a linear time algorithm for the problem.