

Homework 1 - Algorithm Design

Sapienza University of Rome

Marco Costa, 1691388

November 24, 2019

Exercise 1

Problem

Solution

Let's define a matrix M ($k \times n$), in which we have the expected value opening the i^{th} box having j as current best reward ($i \in [0, k-1]$, $j \in [0, n]$)

Algorithm 1 Populate the matrix

```

1: for  $i$  in  $(0, \dots, k-1)$  do
2:   for  $j$  in  $(0, \dots, n)$  do
3:      $M[i, j] \leftarrow -\sum_{h=1}^i c_h + \frac{j}{n+1}(j-1) + \frac{1}{n+1} \sum_{h=j}^n h$ 

```

Now, for each box we can calculate the "a priori" expected value as follow:

$$\bar{v}_k = \sum_{i=0}^n \mathbf{P}_{k-1}(\mathbf{i}) \mathbf{v}_k(\mathbf{i})$$

where $\mathbf{P}_{k-1}(\mathbf{i})$ is the probability of having reward i at least in one of the $k-1$ boxes, while $\mathbf{v}_k(\mathbf{i})$ is the expected value opening the k^{th} box having as current best reward i .

$$\mathbf{P}_{k-1}(\mathbf{i}) = \left(\frac{1}{n+1}\right) \left(\frac{i+1}{n+1}\right)^{k-2} (k-1), \quad \mathbf{v}_k(\mathbf{i}) = M[k, i]$$

So we have:

$$\bar{v}_k = \sum_{i=0}^n \left(\frac{1}{n+1}\right) \left(\frac{i+1}{n+1}\right)^{k-2} (k-1) M[k, i] = \frac{k-1}{n+1} \sum_{i=0}^n \left(\frac{i+1}{n+1}\right)^{k-2} M[k, i]$$

Now we have to calculate \bar{v}_k for each box and return the max:

Algorithm 2 Find expected optimal reward

```

1:  $k_{max} \leftarrow 0$ 
2:  $v[k] \leftarrow \emptyset$ 
3: for  $i$  in  $(0, \dots, k-1)$  do
4:   for  $j$  in  $(0, \dots, n)$  do
5:      $v[i] += \frac{k-1}{n+1} \left(\frac{i+1}{n+1}\right)^{k-2} M[i, j]$ 
6:   if  $v[i] > v[k_{max}]$  then
7:      $k_{max} \leftarrow i$ 
8: return  $v[k_{max}]$ 

```

Exercise 2

First problem

Given a weighted tree T with n nodes, find the complete graph G of minimum weight such that $T \subseteq G$ and T is the unique minimum spanning tree of G .

Solution

Idea: Insert edges that are not in the tree so as to obtain the complete graph. These edges must have a greater weight than those of the tree, so that T is the only MST of G .

Algorithm 3 Find complete graph

```

1: for  $u$  in  $V$  do
2:   for  $v$  in  $V$  do
3:      $e \leftarrow (u, v)$ 
4:     if  $e$  not in  $E$  then
5:        $weight \leftarrow \max(u.getHeaviestEdge().getWeight(), v.getHeaviestEdge().getWeight())$ 
6:        $e.setWeight(weight + 1)$ 
7:        $G.addEdge(e)$ 

```

Cost: $\mathbf{O}(|E|)$, but $|E| = \frac{|V|(|V|-1)}{2}$ and $|V| = n$, so $|E| = \frac{n^2-n}{2}$ and the cost is $\mathbf{O}(n^2)$

Second problem

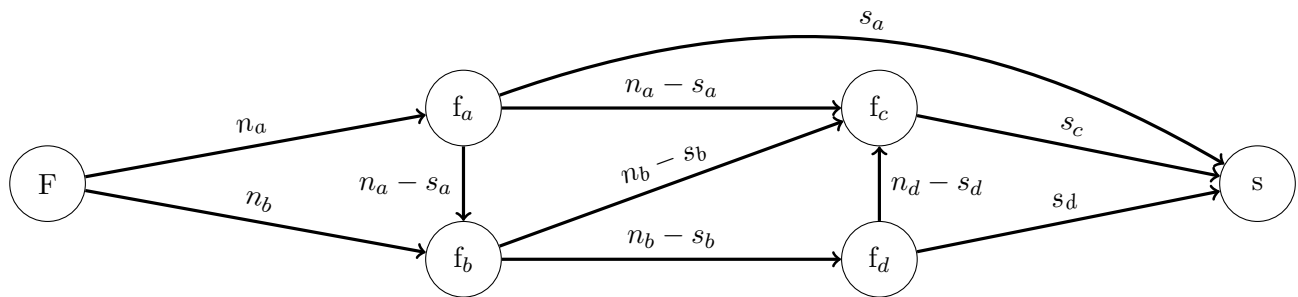
Solution

Hint: Since *Kruskal's* algorithm uses the *Union-Find* structures for representing the cuts, we use this structures to solve the problem.

Exercise 3

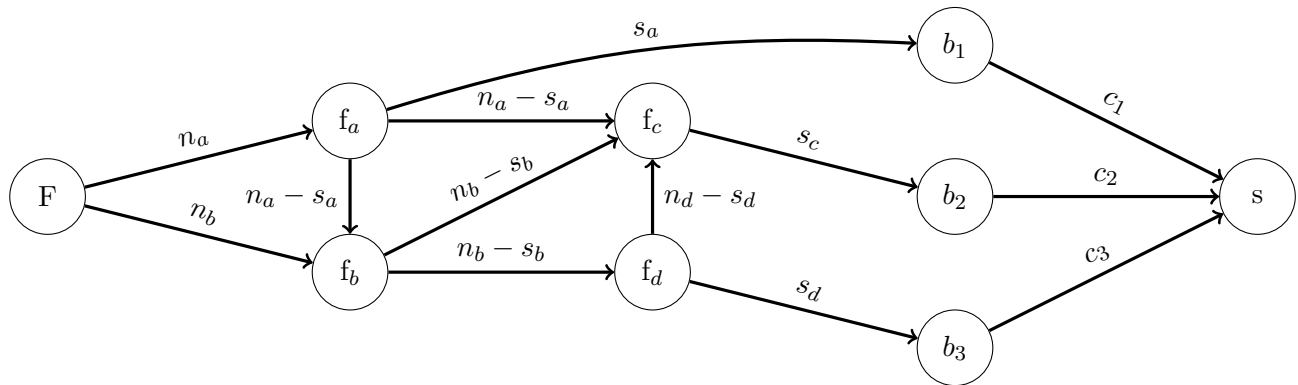
First Problem

Solution



Second Problem

Solution



Exercise 4

Problem

Solution