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

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

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

$$\mathbf{\bar{v_k}} = \sum_{\mathbf{i=0}}^{\mathbf{n}} \mathbf{P_{k-1}(i) v_k(i)}$$

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

$$\mathbf{P_{k-1}(i)} = \left(\frac{\mathbf{1}}{\mathbf{n+1}}\right)\left(\frac{\mathbf{i+1}}{\mathbf{n+1}}\right)^{\mathbf{k-2}}(\mathbf{k-1}), \qquad \mathbf{v_k(i) = M[k,i]}$$

So we have:

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

Now we have to calculate $\mathbf{\bar{v_k}}$ for each box and return the max:

Algorithm 2 Find expected optimal reward

$k_{max} \leftarrow 0$
$v[k] \leftarrow \emptyset$
for i in $(0, ..., k-1)$ **do**
  for j in $(0, ..., n)$ **do**
    $v[i]$ += $\frac{k-1}{n+1}\left(\frac{i+1}{n+1}\right)^{k-2} M[k,i]$
  if $v[i] > v[k_{max}]$ **then**
    $k_{max} \leftarrow i$
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.
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.
Hint: Since *Kruskal*'s algorithm uses the *Union-Find* structures for representing the cuts, we use this structures to solve the problem.

Solution

Algorithm 3 Find complete graph

```
1: for u in set_1 do
2:     for v in set_2 do
3:         if e_new = (u, v) not in E then
4:             e_new.setWeight(e.getWeight() + 1)
5:             G.addEdge(e_new)
```

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 $\mathbf{O}(n^2)$

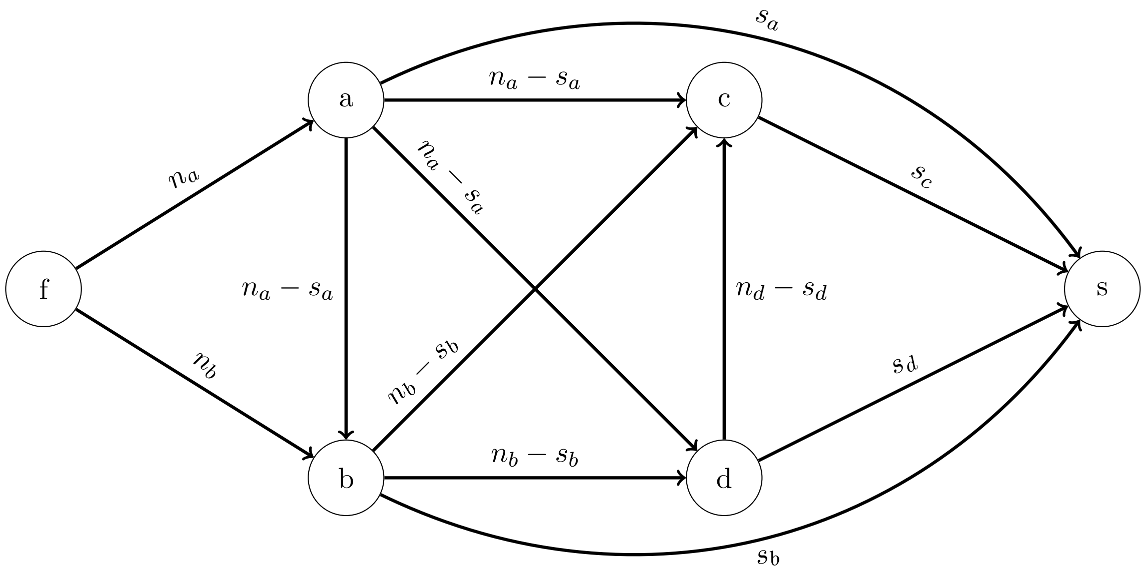
Second problem

Solution

Exercise 3

First Problem

Solution



Second Problem

Solution

Exercise 4

Problem

Solution