CS345 Theoretical Assignment 4

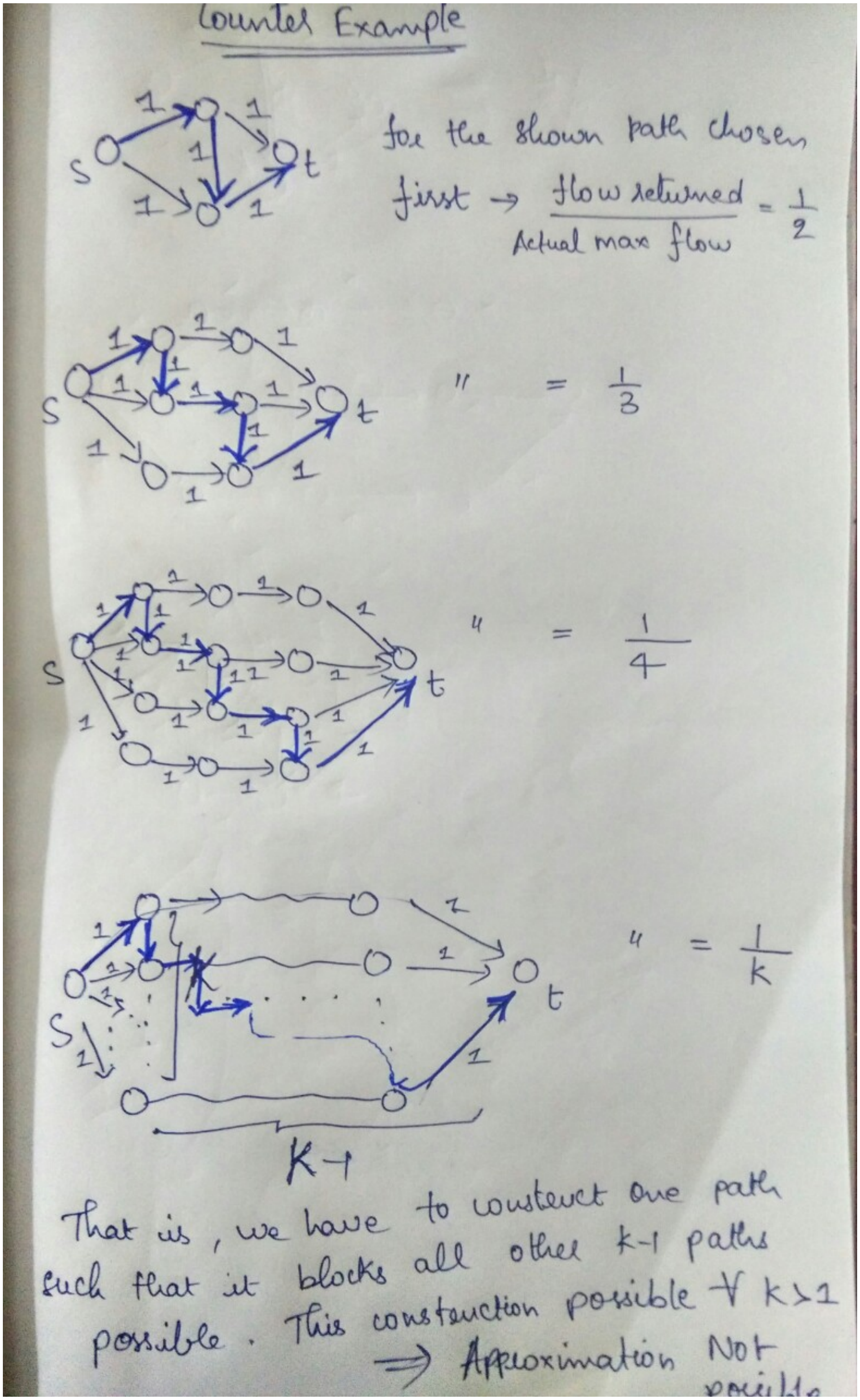
Ayush Agarwal, 13180

M.Arunothia, 13378

Contents

1 Any Guarantee of our First-Attempt Algorithm

1.1 Counter Example



2 Locating faults in a network

2.1 Overview

First we find k-edge disjoint paths of the given graph using the max-flow application discussed in class. Call these $\{p_1, p_2, .., p_k\}$. For all i, we pick p_i and ping its middle vertex. If the result is true, we know that all the vertices to the left of this vertex will also be connected to S. If the result is false, we know that all the vertices to the right of this vertex will also be disconnected from S. Hence, at the end we get vertices marked connected/disconnected from vertex S. **Notice only $O(klog(n))$ pings are issued in the process.**

2.2 Pseudo-Code

```
procedure findConnected(V, E, S, T)
    {p_1, p_2, .., p_k} ← k-disjoint paths between s − t in (V, E)
    S = ϕ
    for i in {1, .., k} do
        Nodes ← nodes lying on path p_i in the order of s − t
        while Nodes <> NULL do
            if ping(Nodes[mid]) == TRUE then
                S = S ∪ Nodes[1..mid]
                Nodes = Nodes[mid + 1..last]
            else
                Nodes = Nodes[1..mid − 1]
            end if
        end while
    end for
    return S
end procedure
```

2.3 Proof of Correctness

2.3.1 Claim: Every disjoint path has exactly one defaulted edge

2.3.2 Arguments

- We are given k edge-disjoint paths from $s - t$.
- It is also given that there is no functional path from $s - t$.
- Hence, every edge-disjoint path should have atleast one defaulted edge otherwise $s - t$ will become connected which will be a contradiction.
- As there are exactly k defaulted edges in total, from above we get that every disjoint path has exactly one defaulted edge.

2.3.3 Claim: Along any edge-disjoint path ping command will produce True,..,True, False,..False

2.3.4 Arguments

- Any path starts with s where $ping(s) = True$ is given.
- Any path ends with t where $ping(t) = False$ is given.
- If the above said output is not produced by ping commands along a path then, there should be two distinct consecutive pairs of nodes in the path whose ping command should produce True, False.
- A True,False pair can occur consecutively only if these two vertices have a defaulted edge connecting them.
- Therefore, if 3 happened then there should be two defaulted edges in a given path, which is a contradiction from the previous claim. This proved this claim.

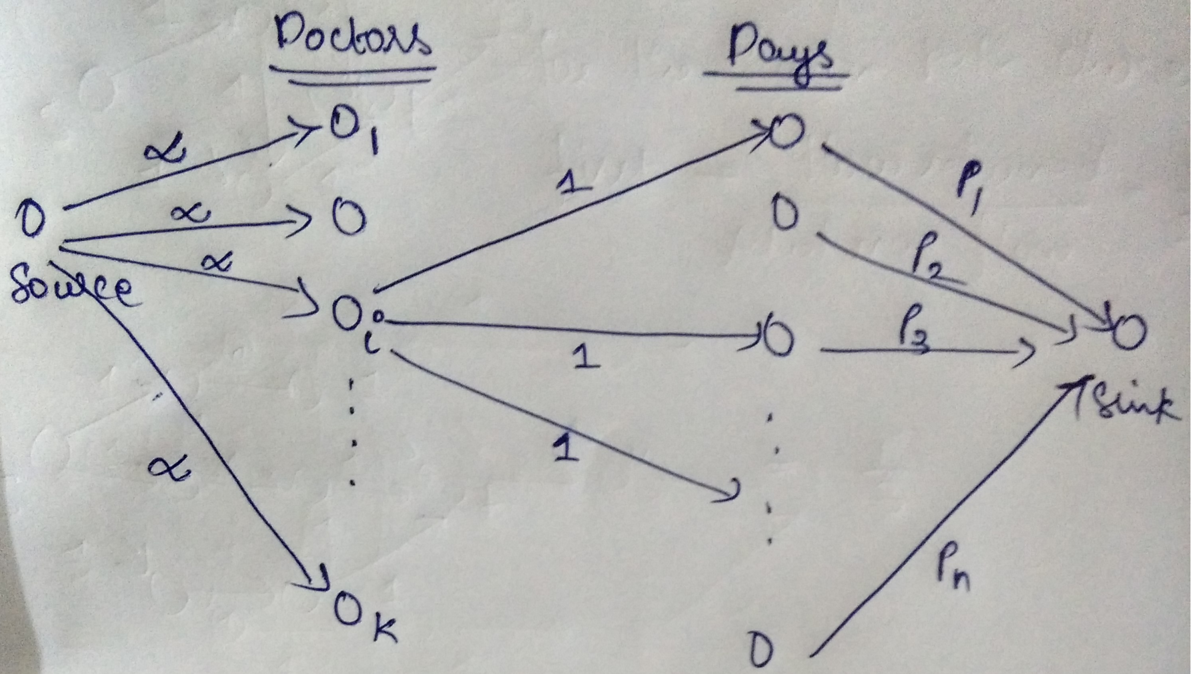
From the previous claim we have the ping pattern True,..,True,False,..,False along any path. This ensures the termination as well as the correctness of the binary search along any path.

3 A max flow application

3.1 Without Extra Constraint

3.1.1 Overview

A max flow Application



Doctors[i] to Days[j] there is an edge iff j lies in the list L_i.

All these edges inserted between Doctors & Days hold capacity 1.

$L_1', \ldots, L_k'$ exists iff Max flow of above graph is $\sum_{i=1}^{n} P_i$.

3.1.2 Notations

p_i denotes the exact number of doctors required on day i
L_i denotes the list of days where doctor i is available
L'_i denotes the list of days that doctor i has to work to produce the required match. Note, $L'_i \subseteq L_i$
$D = \sum_{i=1}^{n} p_i$
n = Number of days in total
k = Number of Doctors in total

3.1.3 Claim

Construction of L'_is is possible if and only if the max-flow in the source-sink graph constructed (in image) is D.

3.1.4 Proof - Part 1

Given L'_is list for all doctors, show that the max flow of source-sink graph shown above is D
Construct a flow f of the source-sink graph as follows.
$f(source, Doctor[i])$ is $|L'_i|$ - satisfies capacity constraint as these edges had infinite capacity
$f(Doctor[i], Day[j])$ is 1 if j lies in L'_i, otherwise is 0 - satisfies capacity constraint
$f(Day[j], sink)$ is p_j - satisfies capacity constraint
Flow Conservation is ensured as it is given to us that L'_is of such definitions exist.
Hence, the above is a valid flow.
As the $cut - capacity$ between $sink$ and the rest of the graph is D, by $min - cut - max - flow$ theorem, f should be a max flow of the source-sink graph. Hence, proved.

3.1.5 Proof - Part 2

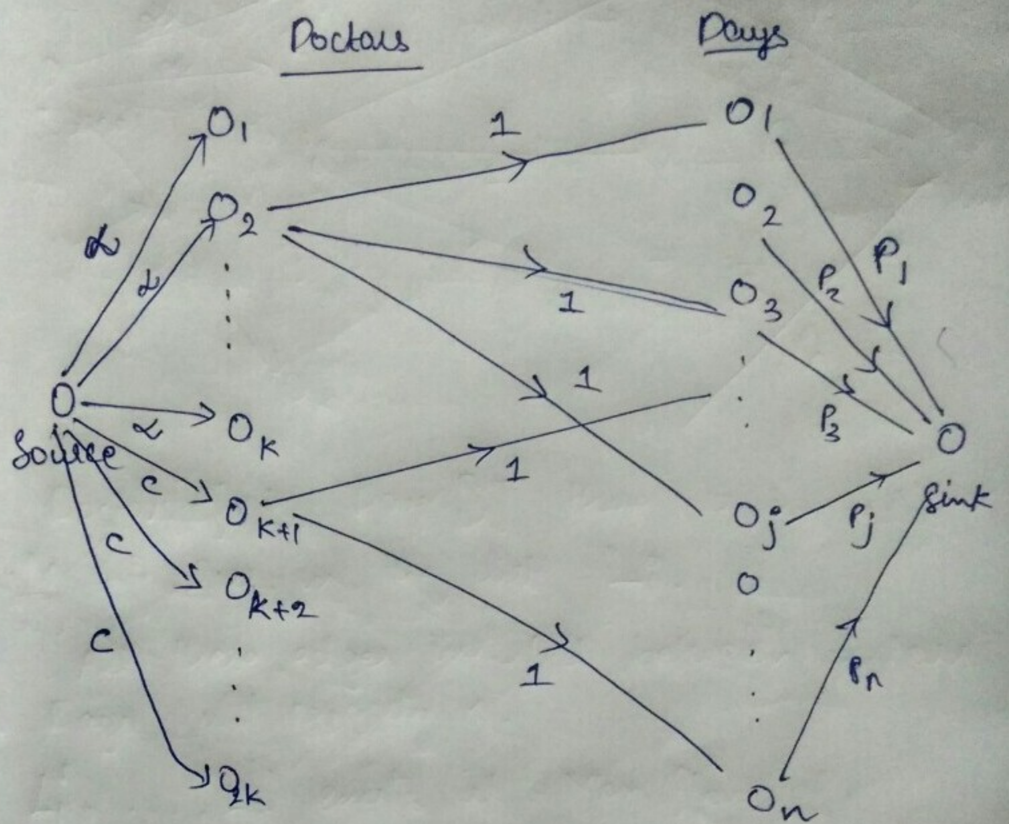
Given the max flow of source-sink graph to be D, show that L'_is list for all doctors exist
Let f be the integral max-flow of the source-sink graph. (Note: Integral flow exists was proved in class)
Construct L'_i as follows.
If $f(Doctor[i], Day[j])$ is 1 then add j to L'_i otherwise do nothing.
Note: $f(Doctor[i], Day[j])$ can be only 0 or 1 by integral flow property.
The L'_is thus constructed are valid as $value(f) = D$ implying every day j has got the exact number of doctors wanted (p_j). Moreover, $L'_i \subseteq L_i$ is ensured from the construction of the source-sink graph itself. Hence, proved.

3.2 With Extra Constraint

3.2.1 Overview

A max flow Application

With Constraint



$$\text{capacity}(\text{Source}, \text{Doctor}[i]) = \begin{cases} \alpha, & 1 \le i \le k \\ c, & k+1 \le i \le 2k \end{cases}$$

$$\text{Capacity}(\text{Doctor}[i], \text{Days}[j]) = \begin{cases} 1, & 1 \le i \le k \ \& \ j \in L_i \\ 0, & 1 \le i \le k \ \& \ j \notin L_i \\ 1, & k+1 \le i \le 2k \ \& \ j \notin L_{\{i-k\}} \\ 0, & k+1 \le i \le 2k \ \& \ j \in L_{\{i-k\}} \end{cases}$$

$\text{Capacity} = 0 \Rightarrow$ No edge.

3.2.2 Notations

p_i denotes the exact number of doctors required on day i
L_i denotes the list of days where doctor i is available
L'_i denotes the list of days that doctor i has to work to produce the required match.
$K_i = L_i \cap L'_i$
$C_i = L'_i - K_i$
$D = \sum_{i=1}^{n} p_i$
n = Number of days in total
k = Number of Doctors in total

3.2.3 Claim

Construction of L'_is is possible if and only if the max-flow in the source-sink graph constructed (in image) is D.

3.2.4 Proof - Part 1

Given L'_is list for all doctors, show that the max flow of source-sink graph shown above is D
Construct a flow f of the source-sink graph as follows.
If $i \leq k$ then
$f(source, Doctor[i])$ is $|K_i|$ - satisfies capacity constraint as these edges had infinite capacity
$f(Doctor[i], Day[j])$ is 1 if j lies in K_i, otherwise is 0 - satisfies capacity constraint
If $i > k$ then
$f(source, Doctor[i])$ is $|C_i|$ - satisfies capacity constraint by definition of L'_is existence.
$f(Doctor[i], Day[j])$ is 1 if j lies in C_i, otherwise is 0 - satisfies capacity constraint

$f(Day[j], sink)$ is p_j - satisfies capacity constraint

Flow Conservation is ensured as it is given to us that L'_is of such definitions exist.
Hence, the above is a valid flow.
As the $cut - capacity$ between $sink$ and the rest of the graph is D, by $min - cut - max - flow$ theorem, f should be a max flow of the source-sink graph. Hence, proved.

3.2.5 Proof - Part 2

Given the max flow of source-sink graph to be D, show that L'_is list for all doctors exist
Let f be the integral max-flow of the source-sink graph. (Note: Integral flow exists was proved in class)
Construct L'_i as follows.
If $f(Doctor[i], Day[j])$ or $f(Doctor[i+k], Day[j])$ is 1 then add j to L'_i otherwise do nothing.
Note: $f(Doctor[i], Day[j])$ can be only 0 or 1 by integral flow property.
The L'_is thus constructed are valid as $value(f) = D$ implying every day j has got the exact number of doctors wanted (p_j). Moreover, both $f(Doctor[i], Day[j])$ and $f(Doctor[i+k], Day[j])$ will not be together 1 from the construction of source-sink graph itself. Hence, proved.