HW3

October 31, 2024

1 Homework 3 - Design and Analysis of Algorithms - Naiara Alonso Montes

1.1 Problem 1

Find a legal flow from s to t in the following network with upper and lower limits. (You don't have to specificy all the steps in the Ford-Flukerson or Edmonds-Karp algorithm that you are using, but you have to explain the construction and the resulting flow.)

I will write down the steps by Ford-Flukerson algorithm used to proof that there exists a legal flow in the following graph $\tilde{N}$.

DBS * s' -> a -> t', bottleneck = 2, flow = 2 * s' -> b -> t', bottleneck = 1, flow = 3 * s' -> c -> t', bottleneck = 2, flow = 5 * s' -> d -> t', bottleneck = 1, flow = 6 * s' -> t -> s -> a -> t', bottleneck = 1, flow = 7 * s' -> c -> t -> s -> t', bottleneck = 1, flow = 8 * s' -> d -> t -> s -> b -> t', bottleneck = 3, flow = 11

At the end of this iterations the resulting network is:

By a theorem used in class, if all edges from leaving from s' are collapsed, there exists a legal flow in network $\tilde{N}$.

Find a maximum flow in the network in part (a). Show all the minimum cuts.

This is the original graph with the flow found in $\tilde{N}$:

This graph meets the vertex and edges rule, but still it is not maximum flow. we can increase the flow by pushing one unit of flow using edge $\{s, a\}$ and $\{s, b\}$:

With this, all edges leaving s are saturated, so the **maximum flow for this graph is 7**.

A searching for all possible min-cut is tedious, I created this Python program that iterates all possible combinations of nodes in a cut an computes its value. All the min-cuts are display as the result of the iteration. This graph has in total **10 min-cuts**.

[ ]:
```
from itertools import combinations

s = [(0, 0), (0, 2), (1, 5), (0, 0), (0, 0), (0, 0)]
a = [(0, 0), (0, 0), (0, 0), (3, 5), (0, 0), (0, 0)]
b = [(0, 0), (2, 4), (0, 0), (0, 0), (2, 4), (0, 0)]
c = [(0, 0), (0, 0), (0, 0), (0, 0), (2, 5), (0, 3)]
d = [(0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (1, 4)]
t = [(0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0)]

edges = [s, a, b, c, d, t]

MAX_FLOW = 7

nodes = ['s', 'a', 'b', 'c', 'd', 't']


def all_possible_cuts(nodes):
  all_combinations = []
  for r in range(0, len(nodes)):
    for comb in combinations(nodes, r):
      all_combinations.append(comb)
  return all_combinations

def min_cut(edges, nodes):
    all_cuts = all_possible_cuts(nodes)
    min_cuts = []
    min_cut_value = float('inf')

    for cut in all_cuts:
      nodes_index_in_cut = [nodes.index(node) for node in cut]

      nodes_in_cut = [nodes[i] for i in nodes_index_in_cut]

      nodes_index = [nodes.index(node) for node in nodes_in_cut]

      capacity = 0
      lower_bound = 0
      for node in nodes_index:
          columns = [edge[node] for edge in edges]
          row = edges[node]
          for i, edge in enumerate(row):
            if i not in nodes_index: # node not in cut
```

```
                capacity += edge[1]
            for i, column in enumerate(columns):
                if i not in nodes_index: # node not in cut
                    lower_bound += column[0]
        if capacity - lower_bound == MAX_FLOW:
          min_cuts.append(cut)

    return min_cuts

min_cut(edges, nodes)
```

```
[ ]: [('s',),
      ('b',),
      ('s', 'd'),
      ('s', 'a', 't'),
      ('a', 'b', 't'),
      ('b', 'c', 'd'),
      ('s', 'a', 'b', 'd'),
      ('s', 'a', 'c', 't'),
      ('a', 'b', 'c', 't'),
      ('s', 'a', 'b', 'c', 'd')]
```

1.2 Problem 2

(a) What is the minimum number of points we need to use? Explain.

The minimum number of points to be used is **4**, as it is the result of the degree of $C(x) + 1$

$$M_4(\omega) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & 1 & \omega^2 \\ 1 & \omega^3 & \omega^2 & \omega \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \tag{1}$$

$$a = \begin{pmatrix} 1 & 2 & 2 & 0 \end{pmatrix} \tag{2}$$

$$b = \begin{pmatrix} -2 & 1 & 0 & 0 \end{pmatrix} \tag{3}$$

Evaluate $A(x)$ at the complex 4th roots of unity. Show at least one level of recursion.

$$\begin{pmatrix} A(1) & = & A(1) \\ A(i) & = & A(\omega) \\ A(-1) & = & A(\omega^2) \\ A(-i) & = & A(\omega^3) \end{pmatrix} = M(i) \cdot a^T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 2 \\ 0 \end{pmatrix} = \tag{4}$$

$$\begin{pmatrix}1 & 1\\ 1 & -1\\ 1 & 1\\ 1 & -1\end{pmatrix}\cdot\begin{pmatrix}1\\ 2\end{pmatrix}+\begin{pmatrix}1 & 1\\ i & -i\\ -1 & -1\\ -i & i\end{pmatrix}\cdot\begin{pmatrix}2\\ 0\end{pmatrix}=\begin{pmatrix}1\cdot 1+1\cdot 2\\ 1\cdot 1+(-1)\cdot 2\\ 1\cdot 1+1\cdot 2\\ 1\cdot 1+(-1)\cdot 2\end{pmatrix}+\begin{pmatrix}1\cdot 2+1\cdot 0\\ i\cdot 2+(-i)\cdot 0\\ -1\cdot 2+(-1)\cdot 0\\ -i\cdot 2+i\cdot 0\end{pmatrix}=\begin{pmatrix}3\\ -1\\ 3\\ -1\end{pmatrix}+\begin{pmatrix}2\\ 2i\\ -2\\ -2i\end{pmatrix}=\begin{pmatrix}5\\ -1+2i\\ 1\\ -1-2i\end{pmatrix} \tag{5}$$

Evaluate $B(x)$ at the complex 4th roots of unity. Show at least one level of recursion.

$$\begin{pmatrix}B(1) & = & B(1)\\ B(i) & = & B(\omega)\\ B(-1) & = & B(\omega^2)\\ B(-i) & = & B(\omega^3)\end{pmatrix}=M(i)\cdot b^T=\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & 1\end{pmatrix}\cdot\begin{pmatrix}-2\\ 1\\ 0\\ 0\end{pmatrix} \tag{6}$$

$$\begin{pmatrix}1 & 1\\ 1 & -1\\ 1 & 1\\ 1 & -1\end{pmatrix}\cdot\begin{pmatrix}-2\\ 0\end{pmatrix}+\begin{pmatrix}1 & 1\\ i & -i\\ -1 & -1\\ -i & i\end{pmatrix}\cdot\begin{pmatrix}1\\ 0\end{pmatrix}=\begin{pmatrix}1\cdot(-2)+1\cdot 0\\ 1\cdot(-2)+(-1)\cdot 0\\ 1\cdot(-2)+1\cdot 0\\ 1\cdot(-2)+(-1)\cdot 0\end{pmatrix}+\begin{pmatrix}1\cdot 1+1\cdot 0\\ i\cdot 1+(-i)\cdot 0\\ -1\cdot 1+(-1)\cdot 0\\ -i\cdot 1+i\cdot 0\end{pmatrix}=\begin{pmatrix}-2\\ -2\\ -2\\ -2\end{pmatrix}+\begin{pmatrix}1\\ i\\ -1\\ -i\end{pmatrix}=\begin{pmatrix}-1\\ -2+i\\ -3\\ -2-i\end{pmatrix} \tag{7}$$

Compute $C(x)$ at the complex 4th roots of unity

$$\begin{pmatrix}C(1)\\ C(\omega)\\ C(\omega^2)\\ C(\omega^3)\end{pmatrix}=\begin{pmatrix}5 & \cdot & (-1)\\ (-1+2i) & \cdot & (-2+i)\\ 1 & \cdot & (-3)\\ (-1-2i) & \cdot & (-2-i)\end{pmatrix}=\begin{pmatrix}-5\\ -5i\\ -3\\ 5i\end{pmatrix} \tag{8}$$

Find the coefficients of $C(x)$

$$\begin{pmatrix}C(\omega^0)\\ C(\omega^1)\\ C(\omega^2)\\ C(\omega^3)\end{pmatrix}=M_n(\omega)\cdot c^T\Rightarrow c^T=\frac{1}{n}\cdot M_n(\omega^{-1}) \tag{9}$$

$$c^T=\frac{1}{4}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -i & -1 & i\\ 1 & -1 & 1 & -1\\ 1 & i & -1 & -i\end{pmatrix}\cdot\begin{pmatrix}-5\\ -5i\\ -3\\ 5i\end{pmatrix}=\frac{1}{4}\cdot\begin{pmatrix}1 & 1\\ 1 & -1\\ 1 & 1\\ 1 & -1\end{pmatrix}\cdot\begin{pmatrix}-5\\ -3\end{pmatrix}+\begin{pmatrix}1 & 1\\ -i & i\\ -1 & -1\\ i & -i\end{pmatrix}\cdot\begin{pmatrix}-5i\\ 5i\end{pmatrix}=\frac{1}{4}\cdot\begin{pmatrix}-8\\ -2\\ -8\\ -2\end{pmatrix}+\begin{pmatrix}0\\ -10\\ 0\\ 10\end{pmatrix}=\frac{1}{4}\cdot\begin{pmatrix}-8\\ -12\\ -8\\ 8\end{pmatrix} \tag{10}$$

$$C(x)=2x^3-2x^2-3x-2 \tag{11}$$

1.3 Problem 3

The matrix A is a circulant matrix which means that each row is a cyclic shift of the previous row. The product of $A \cdot b^T$ can be interpreted as the circular convolution of the first row of A and the vector b. The structure of A allows to use the Fasr Fourier Transform.

Algorithm 1. Calulate the FFT for the first row of A. 2. Calculate the FFT of b. 3. Multiply the FFT of first row of A and the FFT of b 4. Calculate the inverse FFT of the multiplication result.

Complexity

Each FFT use has a complexity of $O(n \log n)$, we are using 3 so it would be $3 \cdot O(n \log n)$, but the constant is discarded, so the final complexity for the algorithm is $O(n \log n)$.

Correctness

1. By this paper, "multiplying by a circulant matriz is equivalent to … operation called circular convolution".
2. By this other publication, "The circular convolution is itself computed by performing the FFT of the two signals, computing their product and then its inverse FFT".

[ ]:
```
!jupyter nbconvert --to pdf HW3.ipynb
```