Advanced Algorithms and Data Structures
Assignment 2

Isabella Odorico `jkd427` Caroline Kierkegaard `qlj556` Mikkel Willén `bmq419` Fie Hammer `gsr530`

January 5, 2024

Linear programming and optimization

29.1-5

We want to convert the following to slack form:

$$\begin{array}{lrcrcrcl}
\text{max} & 2x_1 & & & + & 6x_3 & & \\
\text{s. t} & x_1 & + & x_2 & - & x_3 & \leq & 7 \\
 & 3x_1 & - & x_2 & & & \geq & 8 \\
 & -x_1 & + & 2x_2 & + & 2x_3 & \leq & 0 \\
 & & & x_1, & x_2, & x_3 & \leq & 0
\end{array}$$

We see that there are no non-negativity constraints since: $x_1, x_2, x_3 \geq 0$. We first set it to standard form, by multiplying with - 1:

$$\begin{array}{rcrcrcr}
2x_1 & & & + & 6x_3 & & \\
x_1 & + & x_2 & - & x_3 & \leq & 7 \\
-3x_1 & + & x_2 & & & \leq & -8 \\
x_1 & - & 2x_2 & - & 2x_3 & \leq & 0
\end{array}$$

Then we isolate the right-hand side constants and the left-hand side variables in order to get the equality constraints:

$$\begin{array}{rcrcrcrcrcrcr}
2x_1 & & & + & 6x_3 & & & & & & & & \\
x_1 & + & x_2 & - & x_3 & + & x_4 & & & & & = & 7 \\
-3x_1 & + & x_2 & & & & & + & x_5 & & & = & -8 \\
x_1 & - & 2x_2 & - & 2x_3 & & & & & + & x_6 & = & 0
\end{array}$$

Then we isolate the slack variables to get:

$$\begin{array}{rcrcrcrcr}
z & = & 0 & + & 2x_1 & + & 0x_2 & - & 6x_3 \\
x_4 & = & 7 & - & x_1 & + & x_2 & + & x_3 \\
x_5 & = & -8 & + & 3x_1 & - & x_2 & + & 0x_3 \\
x_6 & = & 0 & - & x_1 + & + & 2x_2 & + & 2x_3
\end{array}$$

x_1, x_2, x_3 is nonbasic and x_4, x_5, x_6 is basic.

29.2-6

We want to write a linear program that, given a bipartite graph G = (V, E), solves the maximum-bipartite-matching problem, by using maximum flow. Here it is:

$$\text{maximize} \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

$$f_{uv} \leq 1 \quad \text{for each } u, v \in V,$$

$$f_{uv} \geq 0 \quad \text{for each } u, v \in V,$$

$$\sum_{\nu \in V} f_{vu} = \sum_{\nu \in V} f_{u\nu} \quad \text{for each } u \in V - \{s, t\}.$$

29.3-5

We solve the linear program using SIMPLEX. It is already on standard form.

$$\begin{array}{lrcrcr} \text{maximize} & 18x_1 & + & 12.5x_2 & & \\ \text{subject to} & x_1 & + & x_2 & \leq & 20 \\ & x_1 & & & \leq & 12 \\ & & & x_2 & \leq & 16 \\ & x_1, & & x_2 & \leq & 0 \end{array}$$

First, we rewrite it on slack form:

$$\begin{array}{rcrcrcr} z & = & 0 & + & 18x_1 & + & 12.5x_2 \\ x_3 & = & 20 & - & x_1 & - & x_2 \\ x_4 & = & 12 & - & x_1 & & \\ x_5 & = & 16 & & & - & x_2 \end{array}$$

We set all the nonbasic variables to 0 and compute the basic variables

$$x_3 = 20, x_4 = 12, x_5 = 16$$

We compute the objective function $z = 0$. This gives the feasible basic solution

$$(0, 0, 20, 12, 16)$$

We select the nonbasic variable x_1. We then select a basic variable, whose constraint most severly limits x_1. In this case, that is x_4.
So x_1 can be increased to 12 without losing feasibility. The new feasible basic solution is

$$(12, 0, 8, 0, 16)$$

and the new objective value is

$$z = 216$$

The constraint defining x_4 is binding.

We will now do the first pivoting by rewriting the slack form to an equivalent slack form with x_1, x_3 and x_5 as basic variables and the new feasible basic solution.
First, the binding constraint defining x_4 is rewritten so that it has x_1 on its left-hand side. Then all occurences of x_1 in the other constraints and in the objective function are replaced by the right-hand side of the binding constraint.

$$\begin{array}{rclclclcl}
z & = & 0 & + & 18(12 - x_4) & + & 12.5x_2 & & \\
x_3 & = & 20 & - & 12 + x_4 & - & x_2 & & \\
x_5 & = & 16 & & & - & x_2 & & \\
x_1 & = & 12 & & & & & - & x_4
\end{array}$$

$$\begin{array}{rclclcl}
z & = & 216 & + & 12.5x_2 & - & 18x_4 \\
x_3 & = & 8 & - & x_2 & + & x_4 \\
x_5 & = & 16 & - & x_2 & & \\
x_1 & = & 12 & & & - & x_4
\end{array}$$

That was the 1. pivoting.

Then we select another nonbasic variable. This time x_2. We select a basic variable, whose constraint most severly limits x_2. In this case, that is x_3.
So x_2 can be increased to 8 without losing feasibility. The new feasible basic solution is

$$(12, 08, 0, 0, 8)$$

and the new objective value is

$$z = 316$$

The constraint defining x_3 is binding. x_3 is rewritten so that it has x_2 on its left-hand side and then all occurences of x_2 in the other constraints and in the objective function are replaced.

$$\begin{array}{rclclcl}
z & = & 216 & + & 12.5(8 + x_4) & - & 18x_4 \\
x_5 & = & 16 & - & (8 + x_4) & & \\
x_1 & = & 12 & & & - & x_4 \\
x_2 & = & 8 & & & + & x_4
\end{array}$$

$$\begin{array}{rclcl}
z & = & 316 & - & 5.5x_4 \\
x_5 & = & 8 & - & x_4 \\
x_1 & = & 12 & - & x_4 \\
x_2 & = & 8 & + & x_4
\end{array}$$

That was the 2. pivoting. There are no more nonbasic variables, so SIMPLEX terminates. The feasible basic solution

$$(12, 8, 0, 0, 8)$$

is optimal.

29.4-1

We formulate the dual of the linear program given in Exercise 29.3-5. To form the dual, we change the maximization to a minimization, exchange the roles of coefficients on the right-hand sides and the objective function, and replace each less-than-or-equal-to by a greater-than-or-equal-to.

$$\begin{array}{lrcrcrcr}
\text{minimize} & 20y_1 & + & 12y_2 & + & 16y_3 & & \\
\text{subject to} & y_1 & + & y_2 & & & \geq & 18 \\
 & y_1 & & & + & y_3 & \geq & 12.5
\end{array}$$

Randomized Algorithms

Exercise 1

In order to get an upper and a lower bound, we have to take into account that the height of the randomly generated binary tree is taking root in the number of comparisons that RandQS makes.
An upper bound can be found by the amount of comparisons between X with the rank i with the rank $j > i$. Much similarly, a lower bound can be found by the amount of comparisons between X with the rank i and the ranks below $j > i$.
We can then take that logic and write the following:

$$\mathbb{E}[d(x_i)] = \mathbb{E}[\#\text{ancestors to i}] + \mathbb{E}[\#\text{successors to i}]$$

That can then be re-written to this:

$$\mathbb{E}[d(x_i)] = \sum_{i>j} Pr[i \text{ is an ancestor of } j] + \sum_{i<j} Pr[j \text{ is an ancestor of } i]$$

With that formula, we can insert the number of comparisons into:

$$\mathbb{E}[d(x)] = \sum_{i>j} \frac{1}{i+1-j} + \sum_{i>j} \frac{1}{j+1-i}$$

We can then use $K = j - i + 1$ for the first part of the formula and then $r = i - j + 1$ for the secound part:

$$\begin{aligned}
\mathbb{E}[d(x)] &= \sum_{k=1}^{n-j+i} \frac{1}{k} + \sum_{r=1}^{n-i+j} \frac{1}{r} \\
&= H_{n-j+i} + H_{n+j-i} \\
&= \theta(\log n)
\end{aligned}$$

With that, we have now proven that $\mathbb{E}[d(x)] = \theta(\log n)$

Exercise 2

The high probability of success is defined as: $1 - n^{-c} = 0.99$ on slide 14 of the shortened version of lecture 4. This we use together with defining a probability of 99%. Afterwards, we simple move and reduce the different values, and get the result:

$$1 - n^{-c} = 0.99$$

<=>

$$n^{-c} = 0.01$$

<=>

$$\log(n^{-c}) = \log(0.01)$$

<=>

$$-c \cdot \log(n) = \log(0.01)$$

<=>

$$c = -\frac{\log(0.01)}{\log(n)}$$

Exercise 1.2

We suppose that at each step of the min-cut algorithm, instead of choosing a random edge for contraction, we choose two vertices and coalesce them into a single vertex. We now show that there are inputs on which the probability that the modified algorithm finds a min-cut is exponentially small.

Let's look at a graph G. We can split the graph into to subgraphs, which are cliques and has the sizes s and t, respectively. The two subgraphs are connected by one edge, which satisfies a min cut. We want to choose to vertices that respect the min cut, which means that they have to be in the same subgraph.
The probability of picking two vertices in the same subgraph is:

$$\mathbb{P} = \frac{\binom{k}{2}}{\binom{n}{2}}$$

We set the size of the first subgraph to s and the size of the second subgraph to t. This gives us the probability of choosing two verteces in the same subgraph as

$$\mathbb{P} = \frac{\binom{s}{2} + \binom{t}{2}}{\binom{s+t}{2}}$$

The probability of the min-cut not being modified for all contractions up to and including l will then be

$$\mathbb{P} = \prod_{i=0}^{l} \left(\frac{\binom{s_i}{2} + \binom{t_i}{2}}{\binom{s_i + t_i}{2}} \right)$$

where s_i and t_i are the size at the i'th iteration.
We have to do $n-2$ contraction, which we can upper bound by n. If we then assume each subgraph has the same size, we can upper bound the sizes and we can set $s_i = \frac{n}{2}$ and $t_i = \frac{n}{2}$. We then get

$$\mathbb{P} = \prod_{i=0}^{n} \left(\frac{2\binom{\frac{n}{2}}{2}}{\binom{n}{2}} \right)$$

Since the expression does not depend on i we can rewrite and we get

$$\mathbb{P} = \left(\frac{2\binom{\frac{n}{2}}{2}}{\binom{n}{2}} \right)^n$$

Since

$$\binom{\frac{n}{2}}{2} \approx n^2$$

and

$$2\left(\frac{n}{2}\right)^2 \leq n^2$$

then we know

$$2\binom{\frac{n}{2}}{2} \leq \binom{n}{2}$$

which means

$$\frac{2\binom{\frac{n}{2}}{2}}{\binom{n}{2}} \leq 1$$

which means as n increases, the expression decreases, which shows that the probability that the modified algorithm finds a min-cut is exponentially small.

Exercise 1.3

We define an algorithm as:

```
LValgo:
    res = A($\Pi$)
    if res is correct:
        return res
    else
        LValgo
```

The algorithm will run until it has found a solution. Looking at the algorithm, we can see, that the approach is a Bernoulli trial, since it will either succeed or fail. The expected value for success will then be

$$\mathbb{E}[X] = \frac{1}{p}$$

where p is the probability. The cost of a single iteration of the algorithm is $T(n) + t(n)$ and the probability of success is $\gamma(n)$. Inserting this gives us the expected running time

$$\mathbb{E}[X] = \frac{T(n) + t(n)}{\gamma(n)}$$

Dispositions

Disposition of Linear Programming Isabella (jkd427)

- Introduction to Linear programming
  - Explain concept
  - Constraints
  - Standard form
  - Slack form
  - Short example (if time)
- SIMPLEX
  - Introduction to duality
  - Proof of weak duality (if time)

Disposition of Randomized algorithms Isabella (jkd427)

- Introduction to randomized algorithms
  - What is it effective for?
  - Running time
- Las Vegas Algorithms
  - What is it and what guarantees does it have?

  - Quicksort example
  - Running time and proof
- Monte Carlo Algorithms
  - What is it and what guarantees does it have?
  - Min-cut example
  - Running time and proof

Disposition of Linear Programming Fie (gsr530)

- Introduction to Linear programming
  - Introduction
  - Constraints
  - Two different forms
- Simplex
  - Duality
  - Proof

Disposition of Randomized algorithms Fie (gsr530)

- Introduction to randomized algorithms
  - Why?
  - Running time
  - Different approaches:
- Las Vegas
  - Why?
  - Quicksort (example)
  - Proof of running time
- Monte Carlo
  - Why?
  - Min-cut (example)
  - Proof of running time

Linear Programming - Caroline (qlj556)

- Explain the concepts
  - The general LP problem
  - Objective function
  - Constraints
  - Standard form
  - Slack form
- Write example
- Introduce the SIMPLEX algorithm
- Explain and prove weak duality

Randomized Algorithms - Caroline (qlj556)

- Talk about motivation for randomized algorithms
- Discuss difference between Las Vegas and Monte Carlo algorithms
- Draw example-graph
- Explain random Quicksort (Las Vegas)
- Prove the expected running time of random Quicksort
- Explain the min-cut algorithm
- Discuss the expected running time of min-cut (prove, if time permits)

Linear Programming - Mikkel (bmq419)

- Introduction
  - General LP problem
  - Objevtive function
  - Constraints
  - Standard form
  - Slack form
- Example from the book
- Simplex
- Proof of weak duality

Randomized Algorithms - Mikkel (bmq419)

- Introduction
  - Unsolvable Problem
  - Faster running time
- Las Vegas algorithms
  - What is it?
  - Randomized quicksort example
  - Proof and running time of randomized QS
- Monte Carlo algorithms
  - What is it?
  - Min-Cut example
  - Proof and running time of Min-Cut