Advanced Algorithms and Data Structures
Assignment 2

Leeann Quynh Do (hlp848)
Marc Pedersen (wfj327)
Jacob Herbst (mwr148)
Jacob Olesen (slc458)

October 5, 2021

Linear programming and optimization

Exercise 29.1-5

We want to convert the following linear program into slack form:

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

We start by isolating the constant on the right and variables on the left in the constraints, such that we get equality constraints.

$$\begin{array}{lrrrrrrrr}
\max & 2x_1 & & +6x_3 & & & & & \\
\text{s.t} & x_1 & +x_2 & -x_3 & +x_4 & & & = & 7 \\
& -3x_1 & +x_2 & & & +x_5 & & = & -8 \\
& x_1 & -2x_2 & -2x_3 & & & +x_6 & = & 0 \\
& x_1, & x_2, & x_3, & x_4, & x_5, & x_6 & \geq & 0
\end{array}$$

we now isolate the slack variables x_4, x_5, x_6 and get the following slack form:

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

The basic variables are the slack variables x_4, x_5, x_6 and the non-basic variables are x_1, x_2, x_3.

Exercise 29.2-6

Write a linear program that, given a bipartite graph $G = (V, E)$, solves the maximum-bipartite-matching problem.
Since we know that the maximum-bipartite-matching problem can be expressed as a max-flow problem we can use the formulation for converting a max flow to LP-problem on p. 860 of CLRS. We however want to tweak it a little to match our problem a little better. For instance we can define our objective function as $\sum_{v \in L} f_{sv}$ since we want to maximize the flow leaving s but because of flow conservation, we only need variables for each edge $(s, v), v \in L$. The first line of constraint defines the flow of each edge to be at most one, since at most one edge in the matching M can be incident on v. The second line of the constraints says that we must preserve flow-conservation, and the last line defines that a flow obviously cannot be negative.

$$
\begin{array}{lrcll}
\max & \displaystyle\sum_{v \in L} f_{sv} & & & \\
\text{s.t} & f_{uv} & \leq & 1 & \text{for each } u, v \in V \\
& \displaystyle\sum_{v \in V} f_{vu} & = & \displaystyle\sum_{v \in V} f_{uv} & \text{for each } u \in L \cup R \quad (\text{same as } V \setminus \{s, t\}) \\
& f_{uv} & \geq & 0 & \text{for each } u, v \in V
\end{array}
$$

Exercise 29.3-5

We will now solve the linear program in Equation 1 using `SIMPLEX`

$$
\begin{array}{lrcrcrl}
\max & 18x_1 & + & 12.5x_2 & & & \\
\text{s.t} & x_1 & + & x_2 & \leq & 20 & \\
& x_1 & & & \leq & 12 & \\
& & & x_2 & \leq & 16 & \\
& x_1 & , & x_2 & \geq & 0 &
\end{array}
\tag{1}
$$

In order to use the simplex algorithm, we must first convert the linear program into slack form. The converted linear program can be seen in Equation 2

$$
\begin{array}{lcrcrcr}
z & = & & 18x_1 & + & 12.5x_2 \\
x_3 & = & 20 & - & x_1 & - & x_2 \\
x_4 & = & 12 & - & x_1 & & \\
x_5 & = & 16 & & & - & x_2
\end{array}
\tag{2}
$$

We now focus on the basic solution where we set all the nonbasic variables to 0, and then we compute the basic variables.
The basic solution:

$$(\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_5) = (0, 0, 20, 12, 16)$$

and the objective value is $z = 0$

We now start the process of pivoting:

1. Pivoting
x_1 can be increased to the value 12 without losing feasibility.

$$\begin{array}{rcrcrcr} z &=& & & 18(12 - x_4) &+& 12.5x_2 \\ x_3 &=& 20 &-& (12 - x_4) &-& x_2 \\ x_1 &=& 12 &-& x_4 & & \\ x_5 &=& 16 & & & - & x_2 \end{array} \tag{3}$$

$$\begin{array}{rcrcrcr} z &=& 216 &-& 18x_4 &+& 12.5x_2 \\ x_3 &=& 8 &+& x_4 &-& x_2 \\ x_1 &=& 12 &-& x_4 & & \\ x_5 &=& 16 & & & - & x_2 \end{array} \tag{4}$$

The new basic variables are $x_1 = 12, x_3 = 8, x_5 = 16$ and the objective value $z = 216$. A feasible solution is $(12, 0, 8, 0, 16)$.

2. Pivoting
x_2 can be increased to the value 8 without losing feasibility.

$$\begin{array}{rcrcrcr} z &=& 216 &-& 18x_4 &+& 12.5(8 + x_4 - x_3) \\ x_2 &=& 8 &+& x_4 &-& x_3 \\ x_1 &=& 12 &-& x_4 & & \\ x_5 &=& 16 & & & - & (8 + x_4 - x_3) \end{array} \tag{5}$$

$$\begin{array}{rcrcrcr} z &=& 316 &-& 5.5x_4 &-& 12.5x_3 \\ x_2 &=& 8 &+& x_4 &-& x_3 \\ x_1 &=& 12 &-& x_4 & & \\ x_5 &=& 8 &-& x_4 &+& x_3 \end{array} \tag{6}$$

Now all coefficients in the objective function are negative, and thus we terminate simplex.

The new basic variables are $x_1 = 12, x_2 = 8, x_5 = 8$. The optimal feasible solution is $(12, 8, 0, 0, 8)$ and the objective value is $z = 316$

Exercise 29.4-1

Formulating the dual of the following linear program:

$$\begin{array}{lrrr} \max & 18x_1 & 12.5x_2 & \\ \text{s.t} & x_1 & +x_2 & \leq 20 \\ & x_1 & & \leq 12 \\ & & x_2 & \leq 16 \\ & & x_1, x_2 & \geq 0 \end{array}$$

Constructing a linear combination of the constraints using nonnegative multipliers y_1, y_2, and y_3:

$$\begin{aligned} y_1(x_1 + x_2) + y_2(x_1) + y_3(x_2) &\leq 20y_1 + 12y_2 + 16y_3 \\ y_1x_1 + y_1x_2 + y_2x_1 + y_3x_2 &\leq 20y_1 + 12y_2 + 16y_3 \\ (y_1 + y_2)x_1 + (y_1 + y_3)x_2 &\leq 20y_1 + 12y_2 + 16y_3 \end{aligned}$$

Now the left-hand side will be an upper bound for the LP if the coefficient at x_1 and x_2 is at least as big as the corresponding coefficient in the objective function:

$$y_1 + y_2 \geq 18, \quad y_1 + y_3 \geq 12.5$$

And now we have to corresponding dual:

$$\begin{array}{lrrrrr} \min & 20y_1 & +12y_2 & +16y_3 & & \\ & y_1 & +y_2 & & \geq & 18 \\ & y_1 & & +y_3 & \geq & 12.5 \\ & & y_1, y_2, y_3 & & \geq & 0 \end{array}$$

Randomized Algorithms

(1)*

When randomized QS runs, the height of the randomly generated binary tree, is based on the number of comparisons that QS makes. An upper bound can be given by looking at the number of comparisons between the specific X with rank i, with rank $j > i$. And a lower bound can be given by looking at the number of comparisons between X with rank i and all ranks below: $i > j$.

This means that we have the following formula:

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

Which can be written as follows:

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

.
The number of comparisons

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

Using $k = j - i + 1$ and $r = i - j + 1$, we have the following:

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

Now we can see that $E[d(x)] = \theta(\log n)$.

(2)*

How many runs of randomized contraction do we need to be 99% sure that a minimum cut is found?
The probability of RandMinCut finding a min-cut is $\geq \frac{2}{n(n-1)}$

Thus we have the following formula:

$$1 - (1 - \frac{2}{n(n-1)})^x = 0.99 \tag{7}$$

By isolating x we have a formula for the number of runs:

$$\begin{aligned}
& 1-(1-\frac{2}{n(n-1)})^x = 0.99 \\
\Leftrightarrow \quad & (1-\frac{2}{n(n-1)})^x = 1-0.99 \\
\Leftrightarrow \quad & \log((1-\frac{2}{n(n-1)})^x) = \log(0.01) \\
\Leftrightarrow \quad & x \cdot \log(1-\frac{2}{n(n-1)}) = \log(0.01) \\
\Leftrightarrow \quad & x = \frac{\log(0.01)}{\log(1-\frac{2}{n(n-1)})}
\end{aligned}$$

(3)*

Exercise 1.2

We now 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.

We start by looking at a graph G, which can be split into two subgraphs, where the number of elements in each subgraph is equal to some k (To ease the calculations, we assume that the 2 subgraphs are the same size). These two subgraphs are connected by an edge that satisfies a min-cut C. We now want to choose two vertices that respect the min-cut, which means that the two vertices must be in the same subgraph.
The probability of choosing two vertices in a subgraph with k vertices are given by:

$$\frac{k(k-1)}{n(n-1)} = \frac{\binom{k}{2}}{\binom{n}{2}} \tag{8}$$

Thus, to get the probability of choosing two vertices in the same subgraph of the two subgraphs, we add these two probabilities:

$$\mathbb{P}(ValidChoice) = \frac{\binom{k}{2} + \binom{k}{2}}{\binom{n}{2}} \tag{9}$$

Which can be rewritten to:

$$\mathbb{P}(ValidChoice) = \frac{2\binom{k}{2}}{\binom{n}{2}} \tag{10}$$

We now define $\mathcal{E}_i$ as an event, where the min-cut C is not modified until the i'th contraction, and the probability for this event is given by

$$\mathbb{P}(\mathcal{E}_i) = \frac{2\binom{k}{2}}{\binom{n-i}{2}} \tag{11}$$

We can then see that, the probability of the min-cut C not being modified for all contractions up to and including i is:

$$\mathbb{P}\left(\mathcal{E}_i | \cap_{l=1}^{i-1} \mathcal{E}_l\right) = \prod_{l=0}^{i} \frac{2\binom{k}{2}}{\binom{n-l}{2}} \tag{12}$$

We now consider an $i = n/4$, and we can upper bound it:

$$\prod_{i=0}^{n/4} \frac{2\binom{k}{2}}{\binom{n-i}{2}} \tag{13}$$

We can now make an upper bound by substituting i with $n/6$:

$$\leq \prod_{i=0}^{n/6} \frac{2\binom{k}{2}}{\binom{n-n/6}{2}} \tag{14}$$

Now the expression is independent of i, and thus we can rewrite the product as follows:

$$= \left(\frac{2\binom{k}{2}}{\binom{n-n/6}{2}} \right)^{n/6} \tag{15}$$

Given that the 2 subsets are equal size, then $k = n/2$. Meaning, we can rewrite and reduce the above term to:

$$= \left(\frac{2\binom{n/2}{2}}{\binom{n-n/6}{2}} \right)^{n/6} \tag{16}$$

$$= \left(\frac{2 \cdot \frac{n}{2}(\frac{n}{2}-1)}{\frac{5n}{4}(\frac{5n}{4}-1)} \right)^{n/6} \tag{17}$$

$$= \left(\frac{\frac{n^2}{2} - n}{\frac{25n^2}{36} - \frac{5n}{6}} \right)^{n/6} \tag{18}$$

$$= \left(\frac{\frac{18n^2}{36} - n}{\frac{25n^2}{36} - \frac{5n}{6}} \right)^{n/6} \tag{19}$$

$$= \left(\frac{\frac{18n^2}{36} - \frac{36n}{36}}{\frac{25n^2}{36} - \frac{25n}{36}} \right)^{n/6} \tag{20}$$

$$= \left(\frac{\frac{18n^2-36n}{36}}{\frac{25n^2-25n}{36}} \right)^{n/6} \tag{21}$$

$$= \left(\frac{648n^2 - 1296n}{900n^2 - 900n} \right)^{n/6} \tag{22}$$

Since the $\lim_{n\to\infty} \left(\frac{648n^2-1296n}{900n^2-900n} \right) = \frac{18}{25}$, we have shown that the probability of the modified algorithm finds a min-cut is exponentially small.

Exercise 1.3

To show how we can obtain a Las Vegas algorithm that always gives us the correct answer given a Monte-Carlo algorithm A for a problem, we first define the algorithm as such:

```
while True:
    solution = A(n)              # Cost T(n)
    if solution is correct:      # Cost t(n)
        return solution
```

We can see that the algorithm will loop forever until it finds a solution, thus constituting a Las Vegas algorithm. It is fairly trivial to see that the approach is a Bernoulli trial, since we either succeed or fail on a given iteration. We can hereby form a geometric distribution of which our random variable X is the probability for success, where the expected value is $\mathbb{E}[X] = \frac{1}{p}$. We know $p = \gamma(n)$ and we can easily multiply our running-time $T(n) + t(n)$ with the expected value, as this is a cost for a single iteration of the loop. Hence we get an expected value of $\mathbb{E}[X] = \frac{T(n)+t(n)}{\gamma(n)}$.

Dispositions

Leeann Quynh Do (hlp848)

Linear programming and optimization

- Introduction to linear programming
  - Objective function and constraints
  - Standard form
  - Slack form
- `SIMPLEX`
- Linear-programming duality

Randomized Algorithms

- Introduction to Randomized Algorithms
  - Motivation
- Las Vegas algorithms
  - Randomized QuickSort
    - Expected running time of `RandQS`
- Monte Carlo algorithm
  - Min-cut example

Marc Pedersen (wfj327)

Linear programming and optimization

- Presentation overview
- concepts and use cases
  - objective function
  - constraints
  - Different algorithms
- Example

- SIMPLEX
- Proof of weak duality

Randomized Algorithms

- Why is random algorithms effective:
  - Unsolvable problems.
  - Faster runningtime.
- Las Vegas Algorithms:
  - What is a Las Vegas algorithm?
  - Randomized Quicksort Example.
    - Proof.
    - Running time.
- Monte Carlo algorithmes:
  - what is a Monte Carlo algorithm?
  - Min-Cut algorithm.
    - Proof.
    - Running time.

Jacob Herbst (mwr148)

Linear programming and optimization

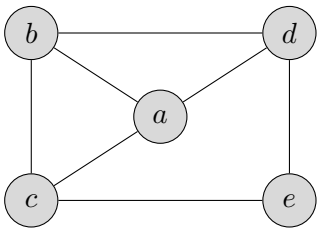
- Presentation outline
- Explain the concepts:
  - general LP problem
  - objective function
  - constraints
  - Standard form
  - slack form
  - write example
- SIMPLEX
- weak duality and proof of it.

example (shamelessly stolen from the book)

$$\begin{array}{lrrl} \min & -2x_1 & +3x_2 & \\ \text{s.t} & x_1 & +x_2 & = 7 \\ & x_1 & -2x_2 & \leq 4 \\ & x_1, & x_2 & \geq 0 \end{array}$$

Randomized Algorithms

- Presentation outline
- Why use randomized algorithms
- explain RandQS, with example? (1,4,5,2,3)
- proof of RandQS expected running time
- Las vegas monte carlo
- random min cut example



Jacob Olesen (slc458)

Linear programming and optimization

- Introduction
- Use cases
- Example
- Simplex
- Duality

Randomized Algorithms

- Introduction
- Use cases and benefits of randomness
- Las Vegas
- Monte Carlo
- Randomized Quick Sort
- Min-cut (if time permits)