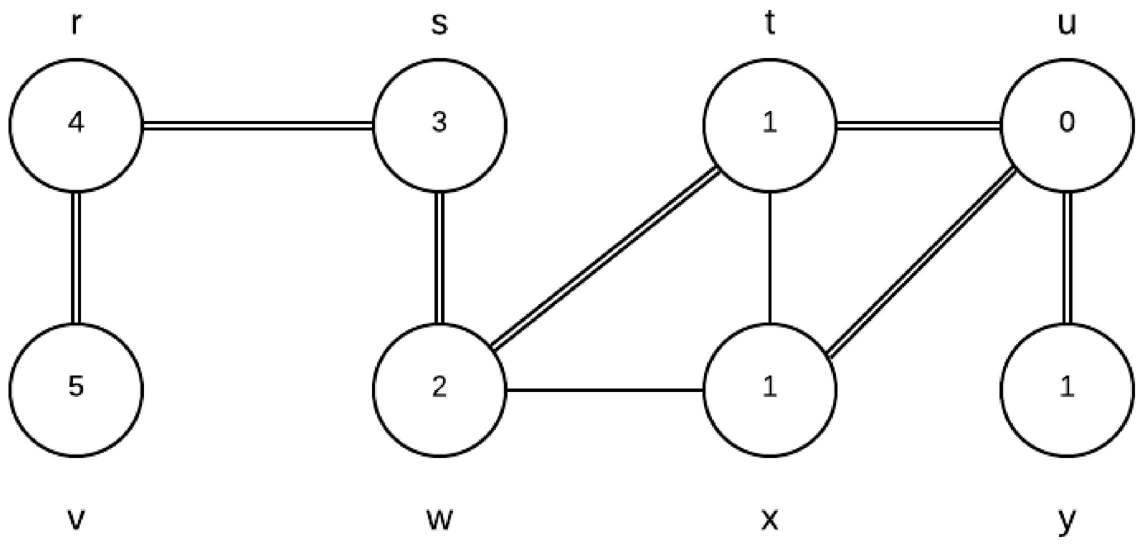
Homework 5

Jeffery Russell

May 9, 2020

1 BFS

(a) CLRS 22.2-2



Vertex	u	t	s	r	v	w	x	y
π	Nil	u	w	s	r	t	u	u
d	0	1	3	4	5	2	1	1

(b) Show that using a single bit to store each vertex color suffices by arguing that the BFS procedure would produce the same result if line 18 were removed.

In the BFS algorithm provided to us, they keep track of the vertex as being white when not-visited, gray when they are visiting their neighbors and black when finished visiting all of the neighbors have been added to the queue. However, the algorithm never actually uses the fact that the color of the vertex has changed to black so it is sufficient to exclude line 18 where the color is set to black. Since this BFS algorithm is only interested if the vertex has been visited yet or not, a single bit would suffice, 0 to represent white and 1 to represent gray.

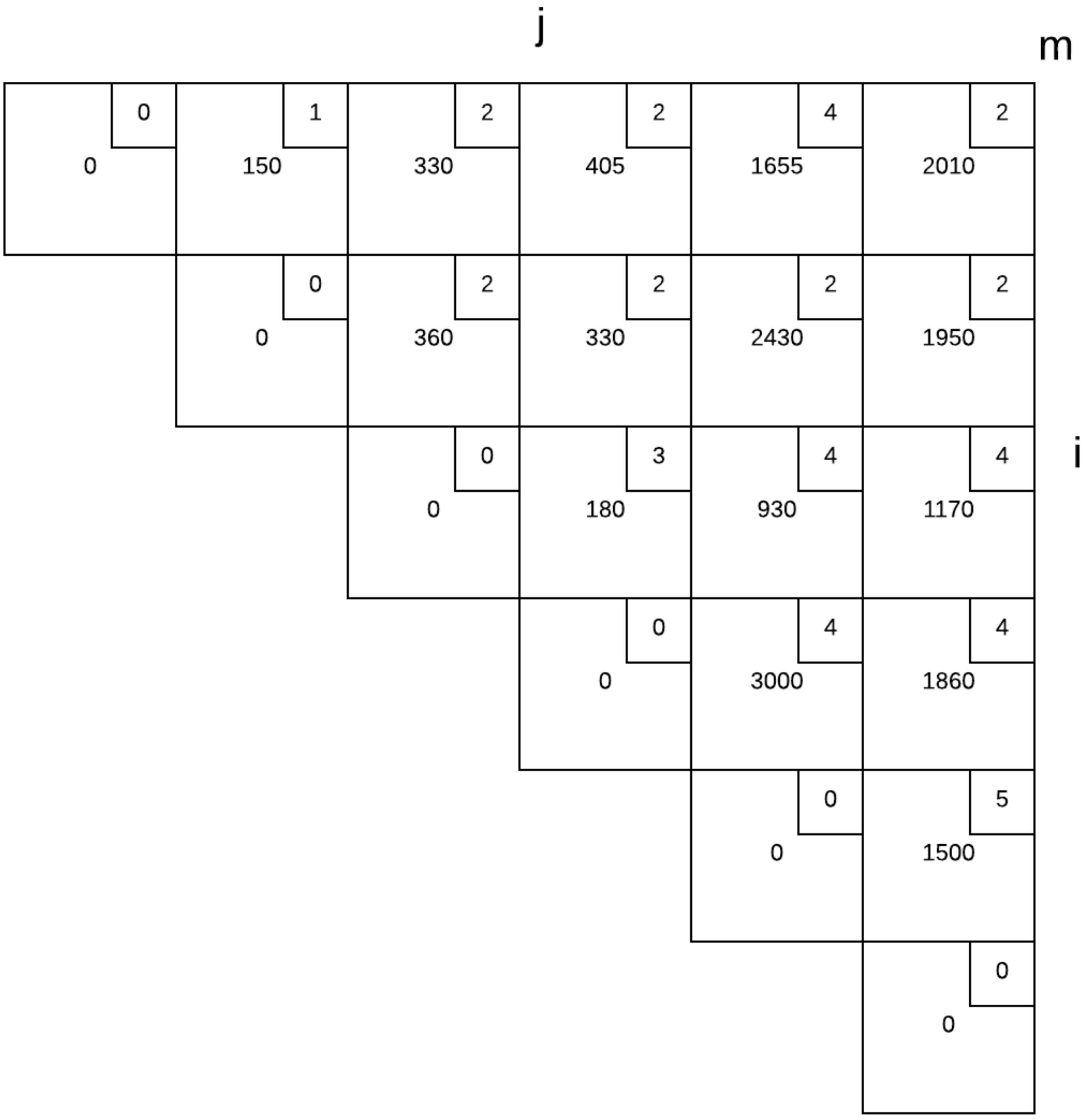
(c) CLRS 22.2-5

The value assigned to d will not change no-matter what order the vertices appear in each adjacency list. However, changing the order of vertices could change the value of the predecessors. Take figure 22.3 for example, to get to vertex w we could have taken either the edge from t or from x. I decided to traverse the adjacent vertices in alphabetical order so I went from t to w and w got a distance of 2 and it's predecessor was set to t. However, if I went from x to x the distance would still be 2, but, it's predecessor would be set to x instead. This property is inherit in the BFS algorithm because it looks for shortest path and does not make any assumptions about the ordering of the adjacency lists.

2 Matrix-Chain Multiplication

Consider the following chain of six matrices: A_1, A_2, A_3, A_4, A_5, and A_6, where A_1 is 5×10, A_2 is 10×3, A_3 is 3×12, A_4 is 12×5, A_5 is 5×50, and A_6 is 50×6. Find an optimal parenthesization of this matrix-chain. Show both the table containing the optimal number of scalar operations for all slices and the choice table.

Answer with 2010 scalar multiplications:
$((\mathrm{A}_1 A_2)((A_3 A_4)(A_5 A_6)))$



$\boxed{0}$ 0	$\boxed{1}$ 150	$\boxed{2}$ 330	$\boxed{2}$ 405	$\boxed{4}$ 1655	$\boxed{2}$ 2010
	$\boxed{0}$ 0	$\boxed{2}$ 360	$\boxed{2}$ 330	$\boxed{2}$ 2430	$\boxed{2}$ 1950
		$\boxed{0}$ 0	$\boxed{3}$ 180	$\boxed{4}$ 930	$\boxed{4}$ 1170
			$\boxed{0}$ 0	$\boxed{4}$ 3000	$\boxed{4}$ 1860
				$\boxed{0}$ 0	$\boxed{5}$ 1500
					$\boxed{0}$ 0

3 Matrix-Chain Multiplication Parenthesize

(a) CLRS 15.3-1
Which is a more efficient way to determine the optimal number of multiplications in a matrix-chain multiplication problem: enumerating all the ways of parenthesizing the product and computing the number of multiplications for each, or running RECURSIVE-MATRIX-CHAIN ? Justify your answer.

Answer
Running the RECURSIVE-MATRIX-CHAIN is more efficient. My argument for this claim is that asymptotically RECURSIVE-MATRIX-CHAIN is faster than enumerating all the solutions.

Enumeration:
As stated in CLS section 15.2, the running time of enumeration is $\Omega(\frac{4^n}{n^{\frac{3}{2}}})$

RECURSIVE-MATRIX-CHAIN:
In section 15-3 of the text book it proved that the RECURSIVE-MATRIX-CHAIN ran in $\Omega(2^n)$. This is better than the lower bound for Enumeration; however, comparing two lower bounds of functions will not help us prove that one actually runs better than the other. To find an upper bound we will reverse the inequality used to find the lower bound.

$T(n) \le 2\sum_{i=1}^{n-1} T(i) + n$

Instead of solving this recurrence with iteration, I will prove that it has an upper bound of $O(3.5^n)$ using induction.

Lemma 1: For any $n \in N$,T(n) $\le 2\sum_{i=1}^{n-1} T(i) + n \le 3.5^n$
Proof via Induction:
BaseCase: n=1

$$LHS = 2\sum_{1}^{1-1} T(i) + 1$$
$$= 1$$
$$RHS = 3.5^1$$
$$= 3.5$$
$$LHS \le RHS$$
$$\square$$

Inductive Step:
Assume proposition is true for all n, show that n+1 follows.

$$LHS = T(n+1) = 2\sum_{i=1}^{n+1-1} T(i) + n + 1$$
$$= 2(1.4(3.5)^n - 1.4) + n + 1 \textit{By I.H}$$
$$= 2.8(3.5)^n - 1.8 + n$$
$$RHS = 3.5^{n+1}$$
$$LHS \le RHS \text{ For sufficiently large n}$$
$$\square$$

Corollary 1: By lemma 1; T(n) is $O(3.5^n)$

Since enumerations runs in $\Omega(\frac{4^n}{n^{\frac{3}{2}}})$ and the recursive solution runs in $O(3.5^n)$, the RECURSIVE-MATRIX-CHAIN is asomtotically faster.

(b) Consider a variant of the matrix-chain multiplication problem in which the goal is to parenthesize the sequence of matrices so as to maximize the number of scalar multiplications. Does this problem exhibit optimal substructure?

Answer
Yes. If we can work to minimize the number of scalar multiplications, we can similarly work to maximize the number of scalar multiplications. A lot like when we worked to use dynamic programming to to minimize the number of multiplications, we can modify our

champion algorithm to instead select the k associated with the maximum amount of scalar multiplications.

4 Text Formatting

Consider the problem of neatly printing a paragraph on the screen (or on a printer). The input text is a sequence S of n words (represented as strings) of lengths $\ell_1, \ldots, \ell_n$ (measured in characters). The input bound M is the maximum number of characters a line can hold. The key to neatly printing a paragraph is to identify in the text sequence the lines of the paragraph so that new-lines can be placed at the end of each line. We can formalize the notion of the "badness" of a line as the number of extra space characters at the end of the line or ∞ if the bound M is exceeded. We can formalize the notion of the "badness" of a paragraph as the badness of the worst (i.e., maximum) line of the paragraph not including the last line. Thus to identify the lines for a neat paragraph, we seek to minimize the badness of the paragraph.

(a) If a given line contains words i through j, and we leave exactly one space between words, the number of extra space characters at the end of the line is $M - j + i - \sum_{k=i}^{j} \ell_k$. Write a mathematical function $es(S, M, i, j)$ that computes the number of extra space characters at the end of a line.

$$es(S, M, i, j) = \begin{cases} M - |S[i]| & \text{if } i = j \\ es(S, M - |S[i]| - 1, i + 1, j) & \text{otherwise} \end{cases}$$

(b) (Project)

(c) Use the function es to write a mathematical function bl(S, M, i, j) that computes line badness.

$$bl(S, M, i, j) = \begin{cases} \infty & \text{if } es(S, M, i, j) < 0 \\ es(S, M, i, j) & \text{otherwise} \end{cases}$$

(d) (Project)

(e) Write a recursive mathematical function mb(S, M) that computes the minimum paragraph badness (using slicing).

$$mb(S, M) = \begin{cases} 0 & \text{if } bl(S, M, 1, |S|) \neq \infty \\ min\{max\{bl(S, M, 1, k), mb(S[k + 1 :], M)\} | & \\ 1 \leq k \leq |S| - 1\} & \text{otherwise} \end{cases}$$

(f) Write a recursive mathematical function mb' (S, M, i) where mb' (S, M, i) = mb(S[i :], M).

$$mb(S,M,i) = \begin{cases} 0 & \text{if } bl(S,M,i,|S|) \neq \infty \\ min\{max\{bl(S,M,i,k), mb'(S,M,k)\}| \\ i+1 \leq k \leq |S|-1\} & \text{otherwise} \end{cases}$$

(g) (Project)

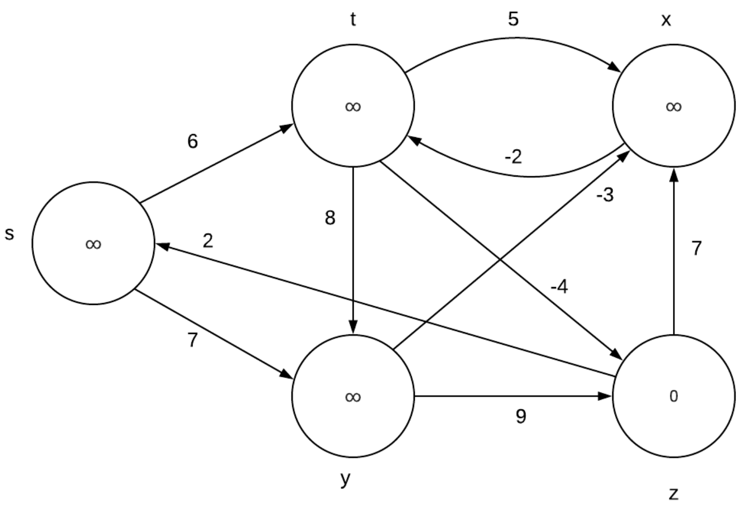
(h) (Project)

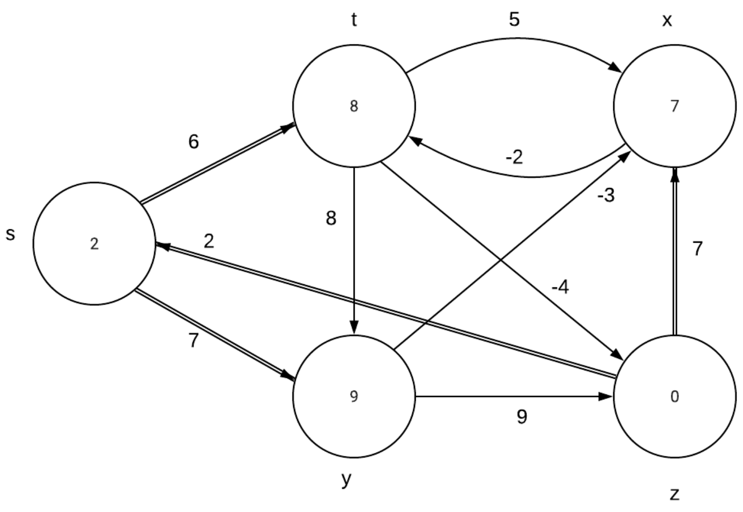
(i) (Project)

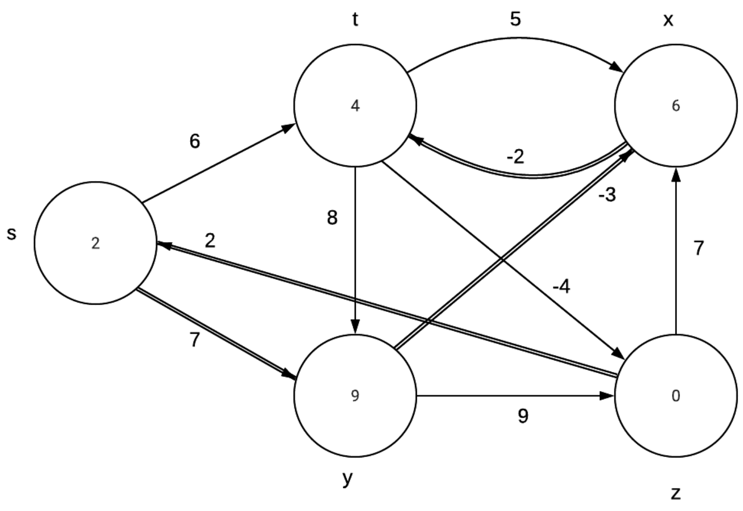
5 Bellman-Ford Algorithm

CLRS 24.1-1
Run the Bellman-Ford algorithm on the directed graph of Figure 24.4, using vertex z as the source. In each pass, relax edges in the same order as in the figure, and show the d and π values after each pass. Now, change the weight of edge (z,x) to 4 and run the algorithm again, using s as the source.

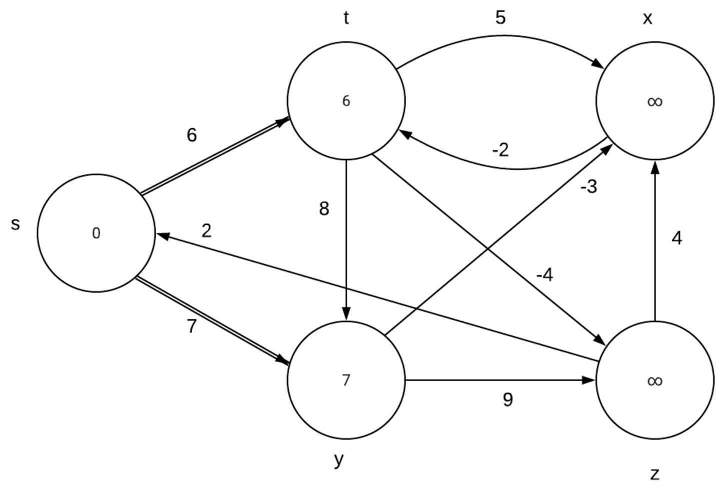
Part a:


t
5
x
∞
∞
6
-2
-3
s
∞
8
2
7
-4
7
∞
0
9
y
z

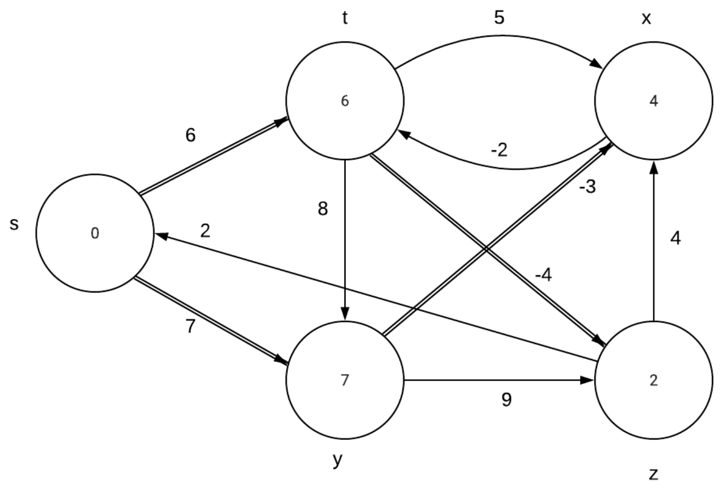

t
5
x
8
7
6
-2
-3
s
2
8
2
7
-4
7
9
0
9
y
z


t
5
x
4
6
6
-2
-3
s
2
8
2
7
-4
7
9
0
9
y
z

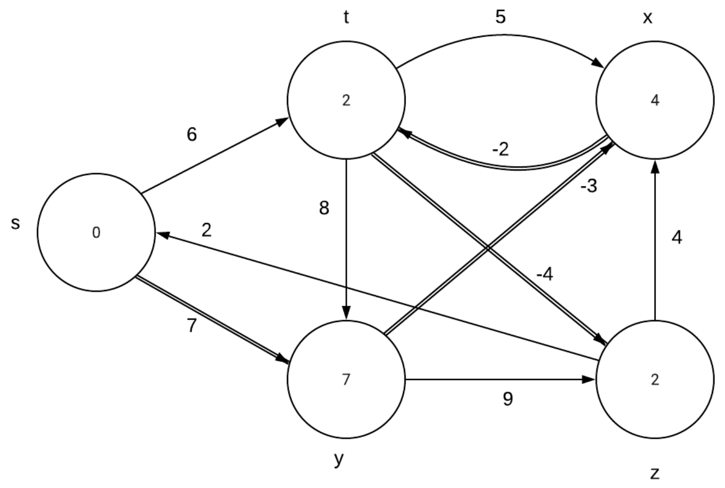
Part b:



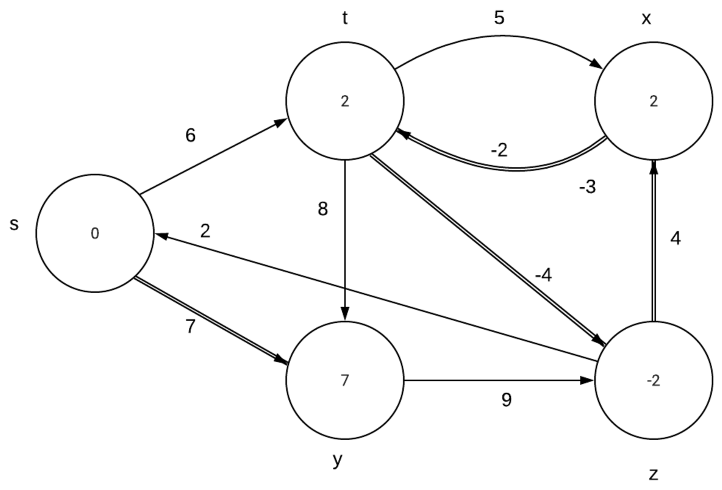
t
5
x
6
6
∞
-2
-3
s
0
8
2
4
-4
7
7
∞
9
y
z



t
5
x
6
6
4
-2
-3
s
0
8
2
4
-4
7
7
2
9
y
z



t
5
x
6
2
4
-2
-3
s
0
8
2
4
-4
7
7
2
9
y
z



t
5
x
6
2
2
-2
-3
s
0
8
2
4
-4
7
7
-2
9
y
z