Assignment 4 - CS 4071 - Spring 2018

Due: 2018-04-04 Group #13: Robert DiMartino (dimartrt), Hayden Schiff (schiffha), Jeremiah Leak (leakjz)

1. Exercise 6.5

Problem: Solve the following instance of the knapsack problem for capacity $C = 30$.

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
v_i	60	50	40	30	20	10	5	1
w_i	30	100	10	10	8	8	1	1

To solve this knapsack problem, we want to greedily select objects of decreasing densities, i.e. the ratio of value to weight, v_i/w_i, until the knapsack is full. So first we compute the densities for each object.

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
$d_i = v_i/w_i$	2	0.5	4	3	2.5	1.25	5	1

Then we repeatedly select available objects with the next highest density until the knapsack is full (or we run out of items).

0. Starting capacity: 30
1. Select: 6 ($d = 5, w = 1$), fraction: 1, remaining capacity: 29
2. Select: 2 ($d = 4, w = 10$), fraction: 1, remaining capacity: 19
3. Select: 3 ($d = 3, w = 10$), fraction: 1, remaining capacity: 9
4. Select: 4 ($d = 2.5, w = 8$), fraction: 1, remaining capacity: 1
5. Select: 0 ($d = 2, w = 30$), fraction: 1/30, remaining capacity: 0

The fractions, f_i, of each item that yield the optimal value for the knapsack are provided in the table below.

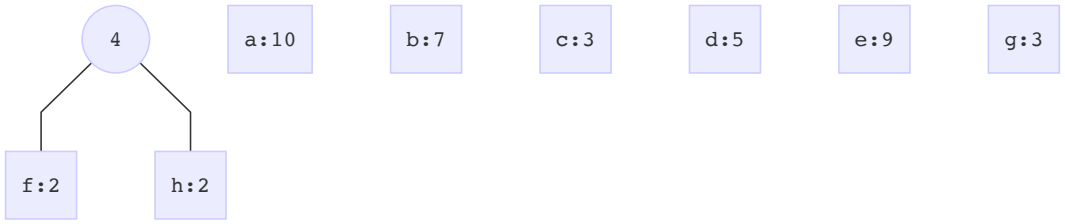
i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Total**
f_i	1/30	0	1	1	1	0	1	0	
$f_i v_i$	2	0	40	30	20	0	5	0	97
$f_i w_i$	1	0	10	10	8	0	1	0	30

2. Exercise 6.9

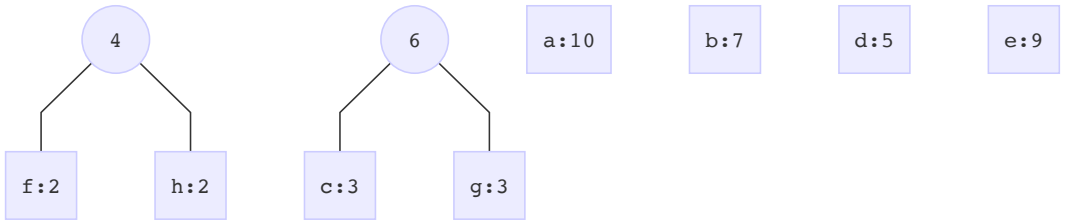
Problem: Trace the action of *HuffmanCode* for the letters *a, b, c, d, e, f, g, h* occuring with frequencies 10, 7, 3, 5, 9, 2, 3, 2.

0. Initial forest

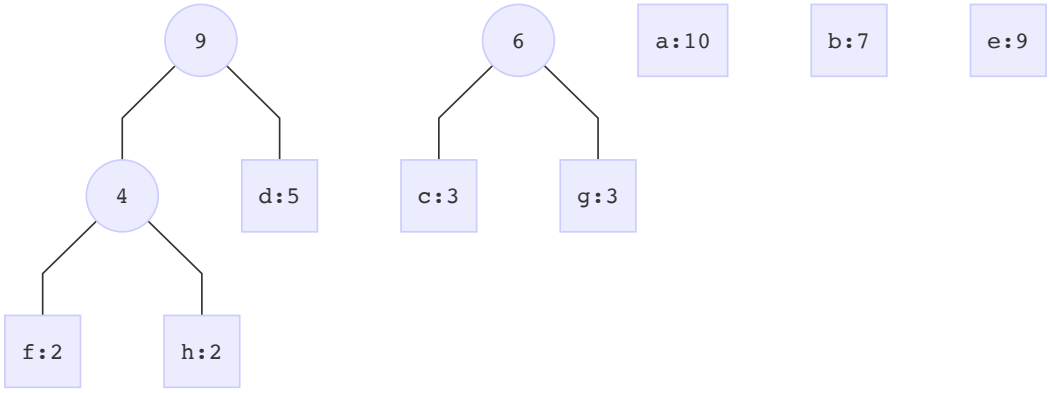
1. Stage 1



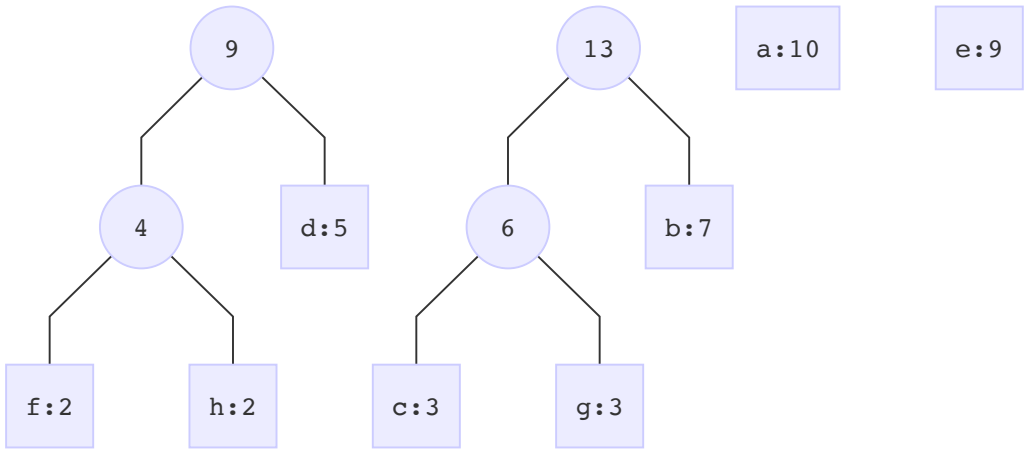
2. Stage 2



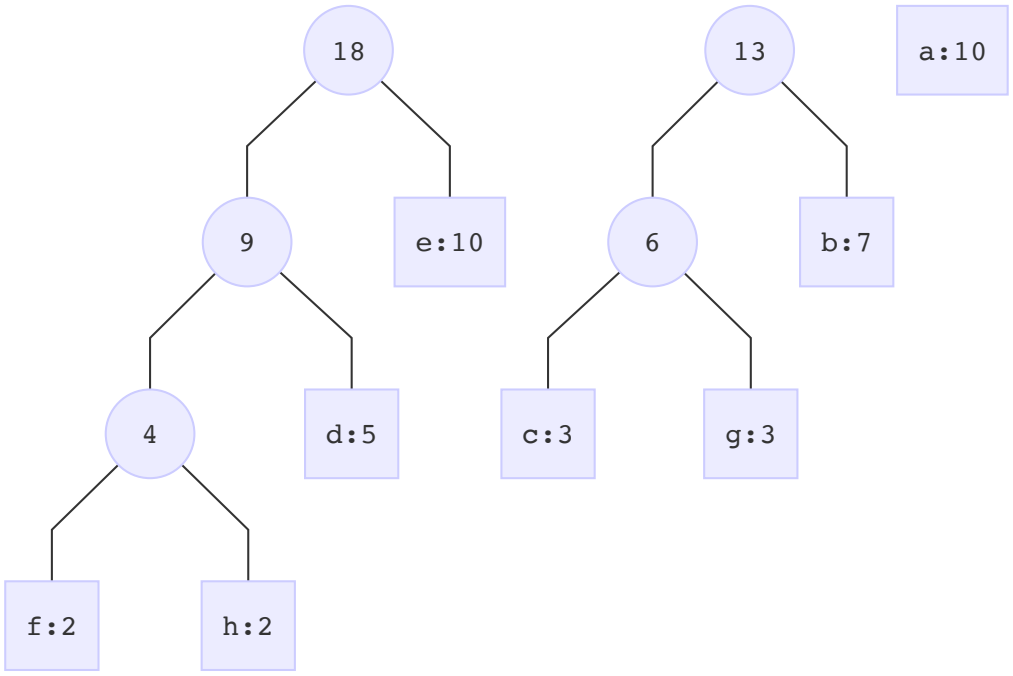
3. Stage 3



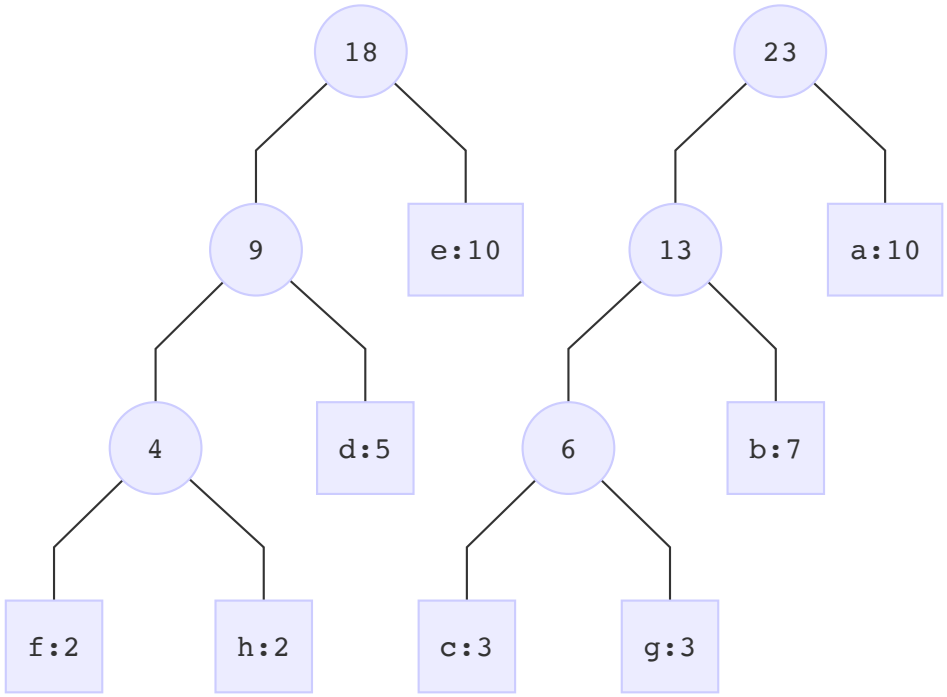
4. Stage 4



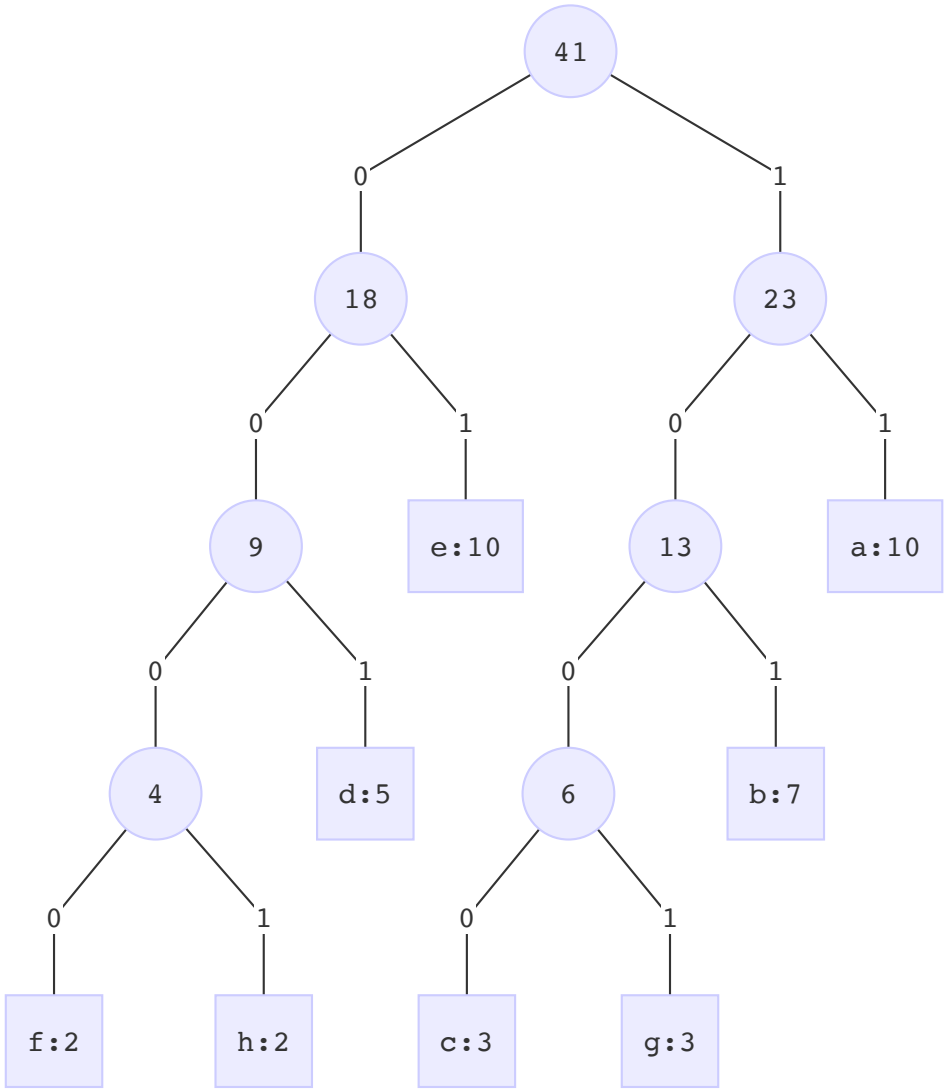
5. Stage 5



6. Stage 6



7. Stage 7: Huffman Tree


41
0
1
18
23
0
1
0
1
9
e:10
13
a:10
0
1
0
1
4
d:5
6
b:7
0
1
0
1
f:2
h:2
c:3
g:3

letter	frequency	encoding
a	10	`11`
b	7	`101`
c	3	`1000`
d	5	`001`
e	9	`01`
f	2	`0000`
g	3	`1001`
h	2	`0001`

3. Exercise 6.10

Problem: Given the Huffman Code Tree in Figure 6.6, decode the string `100111100001101111001`.

> **Figure 6.6**
>
> | letter | frequency | encoding |
> |---|---|---|
> | *a* | 9 | `01` |
> | *b* | 8 | `00` |
> | *c* | 5 | `101` |
> | *d* | 3 | `1001` |
> | *e* | 15 | `11` |
> | *f* | 2 | `1000` |

We read binary characters from the encoded string one at a time. Because the Huffman Code is a prefix code, when the binary substring we've read matches one of the letter encodings from Figure 6.6, we can unambiguously determine which letter is decoded.

Reading the string from the front:

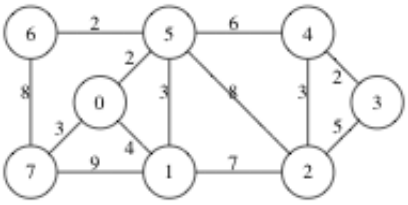
1. `1001` = *d*

2. `11` = *e*
3. `1000` = *f*
4. `01` = *a*
5. `101` = *c*
6. `11` = *e*
7. `1001` = *d*

Then `100111100001101111001` decoded is "defaced".

4.

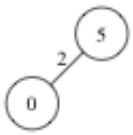
Consider the following weighted graph G.



a)

Problem: Trace the action of procedure `Kruskal` for G.

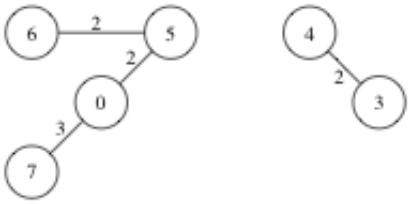
1. Stage 1



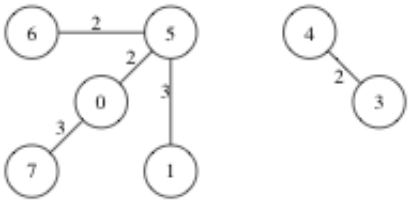
2. Stage 2

3. Stage 3

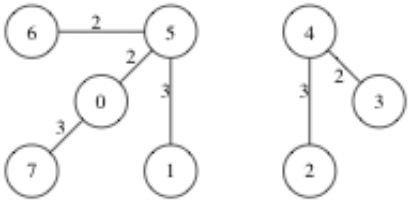
4. Stage 4



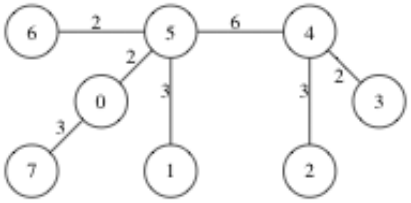
5. Stage 5



6. Stage 6



7. Stage 7



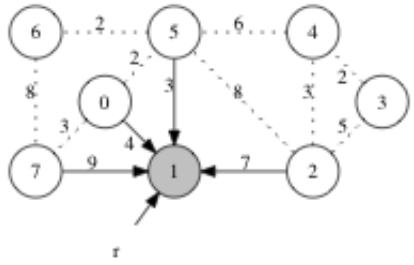
b)

Problem: Trace the action of procedure `Prim` for G, with r = 1.

0. Stage 0

i	0	1	2	3	4	5	6	7
nearest[i]	4	0	7	inf	inf	3	inf	9
parent[i]	1	-1	1	-	-	1	-	1
inTree[i]	F	T	F	F	F	F	F	F

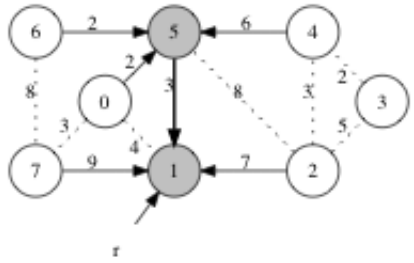
Weight = 0



1. Stage 1

i	0	1	2	3	4	5	6	7
nearest[i]	2	0	7	inf	6	3	2	9
parent[i]	5	-1	1	-	5	1	5	1
inTree[i]	F	T	F	F	F	T	F	F

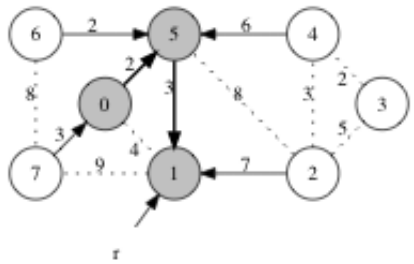
Weight = 3



2. Stage 2

i	0	1	2	3	4	5	6	7
nearest[i]	2	0	7	inf	6	3	2	3
parent[i]	5	-1	1	-	5	1	5	0
inTree[i]	T	T	F	F	F	T	F	F

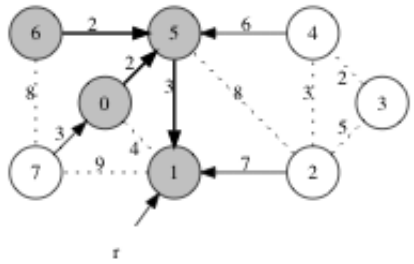
Weight = 5



3. Stage 3

i	0	1	2	3	4	5	6	7
nearest[i]	2	0	7	inf	6	3	2	3
parent[i]	5	-1	1	-	5	1	5	0
inTree[i]	T	T	F	F	F	T	T	F

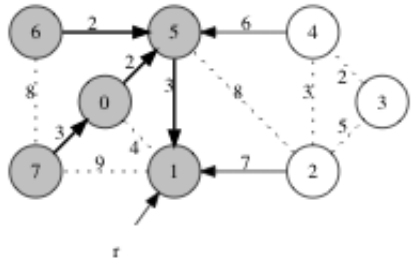
Weight = 7



4. Stage 4

i	0	1	2	3	4	5	6	7
nearest[i]	2	0	7	inf	6	3	2	3
parent[i]	5	-1	1	-	5	1	5	0
inTree[i]	T	T	F	F	F	T	T	T

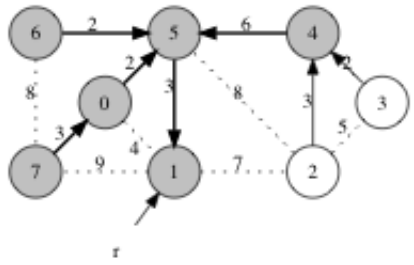
Weight = 10



5. Stage 5

i	0	1	2	3	4	5	6	7
nearest[i]	2	0	3	2	6	3	2	3
parent[i]	5	-1	4	4	5	1	5	0
inTree[i]	T	T	F	F	T	T	T	T

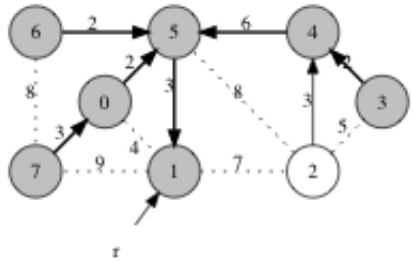
Weight = 16



6. Stage 6

i	0	1	2	3	4	5	6	7
nearest[i]	2	0	3	2	6	3	2	3
parent[i]	5	-1	4	4	5	1	5	0
inTree[i]	T	T	F	T	T	T	T	T

Weight = 18

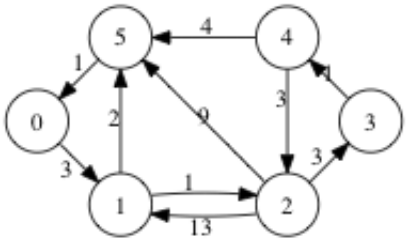


7. Stage 7

i	0	1	2	3	4	5	6	7
nearest[i]	2	0	3	2	6	3	2	3
parent[i]	5	-1	4	4	5	1	5	0
inTree[i]	T	T	T	T	T	T	T	T

Weight = 21

5.

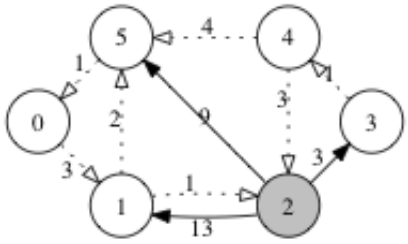


a)

Problem: Trace the action of procedure Dijkstra for the digraph with initial vertex $r = 2$.

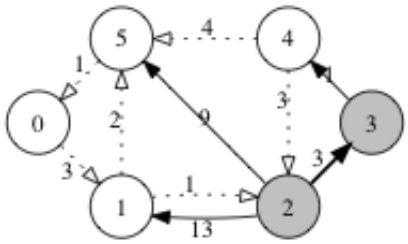
0. Stage 0

i	0	1	2	3	4	5
dist[i]	inf	13	0	3	inf	9
parent[i]	-	2	-1	2	-	2
inTree[i]	F	F	T	F	F	F



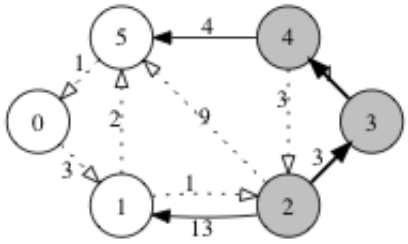
1. Stage 1

i	0	1	2	3	4	5
dist[i]	inf	13	0	3	4	9
parent[i]	-	2	-1	2	3	2
inTree[i]	F	F	T	T	F	F



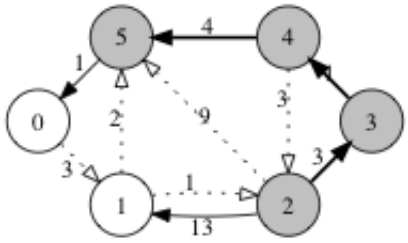
2. Stage 2

i	0	1	2	3	4	5
dist[i]	inf	13	0	3	4	8
parent[i]	-	2	-1	2	3	4
inTree[i]	F	F	T	T	T	F



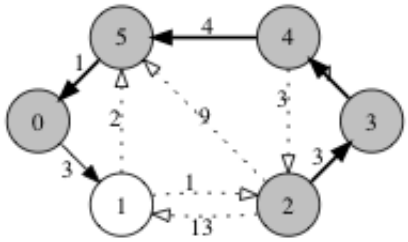
3. Stage 3

i	0	1	2	3	4	5
dist[i]	9	13	0	3	4	8
parent[i]	5	2	-1	2	3	4
inTree[i]	F	F	T	T	T	T



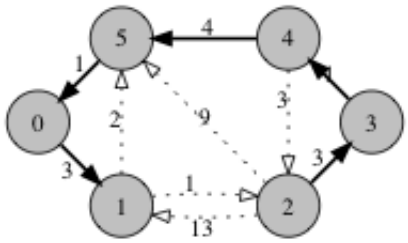
4. Stage 4

i	0	1	2	3	4	5
dist[i]	9	12	0	3	4	8
parent[i]	5	0	-1	2	3	4
inTree[i]	T	F	T	T	T	T

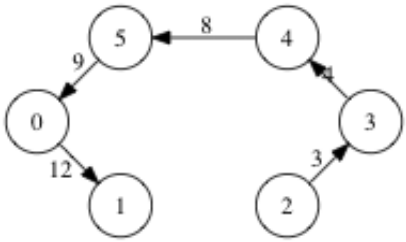


5. Stage 5

i	0	1	2	3	4	5
dist[i]	9	12	0	3	4	8
parent[i]	5	0	-1	2	3	4
inTree[i]	T	T	T	T	T	T



Minimum spanning path:

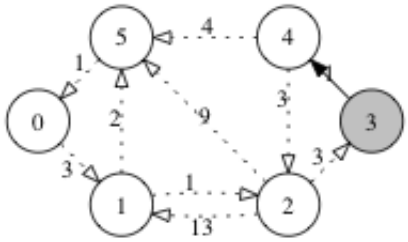


b)

Problem: Repeat for $r = 3$.

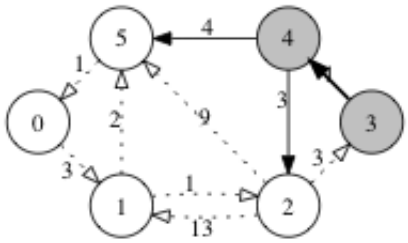
0. Stage 0

i	0	1	2	3	4	5
dist[i]	inf	inf	inf	0	1	inf
parent[i]	-	-	-	-1	3	-
inTree[i]	F	F	F	T	F	F



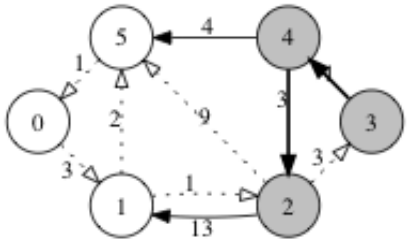
1. Stage 1

i	0	1	2	3	4	5
dist[i]	inf	inf	4	0	1	5
parent[i]	-	-	4	-1	3	4
inTree[i]	F	F	F	T	T	F



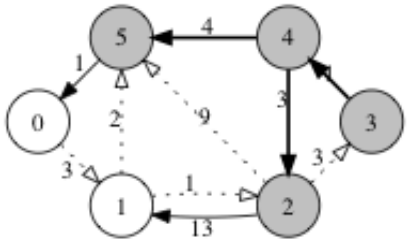
2. Stage 2

i	0	1	2	3	4	5
dist[i]	inf	17	4	0	1	5
parent[i]	-	2	4	-1	3	4
inTree[i]	F	F	T	T	T	F



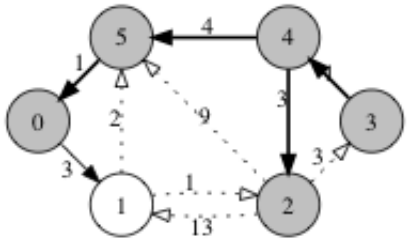
3. Stage 3

i	0	1	2	3	4	5
dist[i]	6	17	4	0	1	5
parent[i]	5	2	4	-1	3	4
inTree[i]	F	F	T	T	T	T



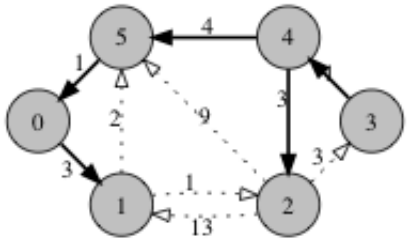
4. Stage 4

i	0	1	2	3	4	5
dist[i]	6	9	4	0	1	5
parent[i]	5	0	4	-1	3	4
inTree[i]	T	F	T	T	T	T

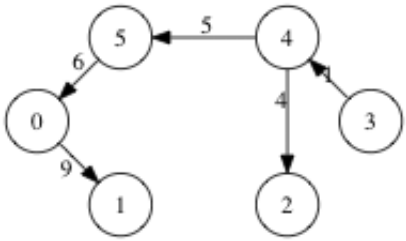


5. Stage 5

i	0	1	2	3	4	5
dist[i]	6	9	4	0	1	5
parent[i]	5	0	4	-1	3	4
inTree[i]	T	T	T	T	T	T



Minimum spanning path:



6. Exercise 7.13

Problem: Verify formula (7.4.4).

$$AB = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} a_{00}b_{00} + a_{01}b_{10} & a_{00}b_{01} + a_{01}b_{11} \\ a_{10}b_{00} + a_{11}b_{10} & a_{10}b_{01} + a_{11}b_{11} \end{bmatrix}$$

$$AB \stackrel{?}{=} \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}$$

$$\implies \begin{aligned} &(1)\ a_{00}b_{00} + a_{01}b_{10} \stackrel{?}{=} m_1 + m_4 - m_5 + m_7 \\ &(2)\ a_{00}b_{01} + a_{01}b_{11} \stackrel{?}{=} m_3 + m_5 \\ &(3)\ a_{10}b_{00} + a_{11}b_{10} \stackrel{?}{=} m_2 + m_4 \\ &(4)\ a_{10}b_{01} + a_{11}b_{11} \stackrel{?}{=} m_1 + m_3 - m_2 + m_6 \end{aligned}$$

To verify 7.4.4, we need to verify each of its four equations, where:

$$\begin{aligned} m_1 &= (a_{00} + a_{11})(b_{00} + b_{11}) &&= a_{00}b_{00} + a_{00}b_{11} + a_{11}b_{00} + a_{11}b_{11} \\ m_2 &= (a_{10} + a_{11})(b_{00}) &&= a_{10}b_{00} + a_{11}b_{00} \\ m_3 &= a_{00}(b_{01} - b_{11}) &&= a_{00}b_{01} - a_{00}b_{11} \\ m_4 &= a_{11}(b_{10} - b_{00}) &&= a_{11}b_{10} - a_{11}b_{00} \\ m_5 &= (a_{00} + a_{01})(b_{11}) &&= a_{00}b_{11} + a_{01}b_{11} \\ m_6 &= (a_{10} - a_{00})(b_{00} + b_{10}) &&= a_{10}b_{00} + a_{10}b_{10} - a_{00}b_{00} - a_{00}b_{10} \\ m_7 &= (a_{01} - a_{11})(b_{10} + b_{11}) &&= a_{01}b_{10} + a_{01}b_{11} - a_{11}b_{10} - a_{11}b_{11} \end{aligned}$$

1.

$$
\begin{gathered}
m_1 + m_4 - m_5 + m_7 \\
= (a_{00}b_{00} + a_{00}b_{11} + a_{11}b_{00} + a_{11}b_{11}) + (a_{11}b_{10} - a_{11}b_{00}) - \dots \\
(a_{00}b_{11} + a_{01}b_{11}) + (a_{01}b_{10} + a_{01}b_{11} - a_{11}b_{10} - a_{11}b_{11}) \\
= a_{00}b_{00} + \cancel{a_{00}b_{11}}^{1} + \cancel{a_{11}b_{00}}^{2} + \cancel{a_{11}b_{11}}^{3} + \cancel{a_{11}b_{10}}^{4} - \cancel{a_{11}b_{00}}^{2} + \dots \\
- \cancel{a_{00}b_{11}}^{1} - \cancel{a_{01}b_{11}}^{5} + a_{01}b_{10} + \cancel{a_{01}b_{11}}^{5} - \cancel{a_{11}b_{10}}^{4} - \cancel{a_{11}b_{11}}^{3} \\
= a_{00}b_{00} + a_{01}b_{10}
\end{gathered}
$$

2.

$$
\begin{aligned}
m_3 + m_5 &= (a_{00}b_{01} - a_{00}b_{11}) + (a_{00}b_{11} + a_{01}b_{11}) \\
&= a_{00}b_{01} - \cancel{a_{00}b_{11}} + \cancel{a_{00}b_{11}} + a_{01}b_{11} \\
&= a_{00}b_{01} + a_{01}b_{11}
\end{aligned}
$$

3.

$$
\begin{aligned}
m_2 + m_4 &= (a_{10}b_{00} + a_{11}b_{00}) + (a_{11}b_{10} - a_{11}b_{00}) \\
&= a_{10}b_{00} + \cancel{a_{11}b_{00}} + a_{11}b_{10} - \cancel{a_{11}b_{00}} \\
&= a_{10}b_{00} + a_{11}b_{10}
\end{aligned}
$$

4.

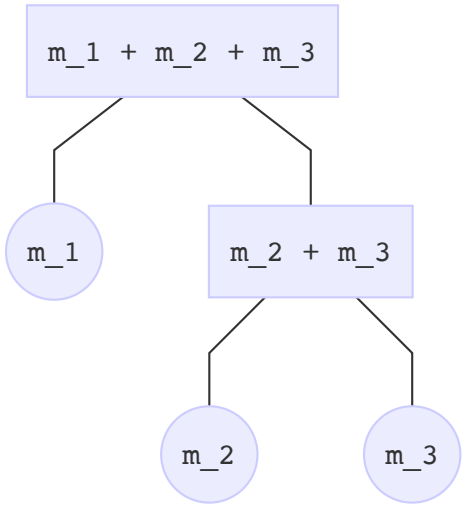
$$
\begin{gathered}
m_1 + m_3 - m_2 + m_6 \\
= (a_{00}b_{00} + a_{00}b_{11} + a_{11}b_{00} + a_{11}b_{11}) + (a_{00}b_{01} - a_{00}b_{11}) + \dots \\
-(a_{10}b_{00} + a_{11}b_{00}) + (a_{10}b_{00} + a_{10}b_{10} - a_{00}b_{00} - a_{00}b_{10}) \\
= \cancel{a_{00}b_{00}}^{1} + \cancel{a_{00}b_{11}}^{2} + \cancel{a_{11}b_{00}}^{3} + a_{11}b_{11} + \cancel{a_{00}b_{01}}^{4} - \cancel{a_{00}b_{11}}^{2} + \dots \\
- \cancel{a_{10}b_{00}}^{5} - \cancel{a_{11}b_{00}}^{3} + \cancel{a_{10}b_{00}}^{5} + a_{10}b_{10} - \cancel{a_{00}b_{00}}^{1} - \cancel{a_{00}b_{10}}^{4} \\
= a_{10}b_{01} + a_{11}b_{11}
\end{gathered}
$$

Since each of the four equations are true, formula 7.4.4 is verified.

7.

Problem: Design a greedy algorithm to solve the optimal merge pattern problem. In this problem, we have n sorted files of lengths $f_0, f_1, \dots, f_{n-1}$, and we wish to merge them into a single file by a sequence of merges of pairs of files. To merge two files of lengths m_1 and m_2 takes $m_1 + m_2$ operations. Describe your algorithm in general, and illustrate it for files of lengths 10,7,3,5,9,2,3,2. (Can you make a connection with Huffman codes and Exercise 1 of this assignment?)

A key consideration is that files that are merged early in the process will continue to contribute to the number of operations needed for successive merges. For example:

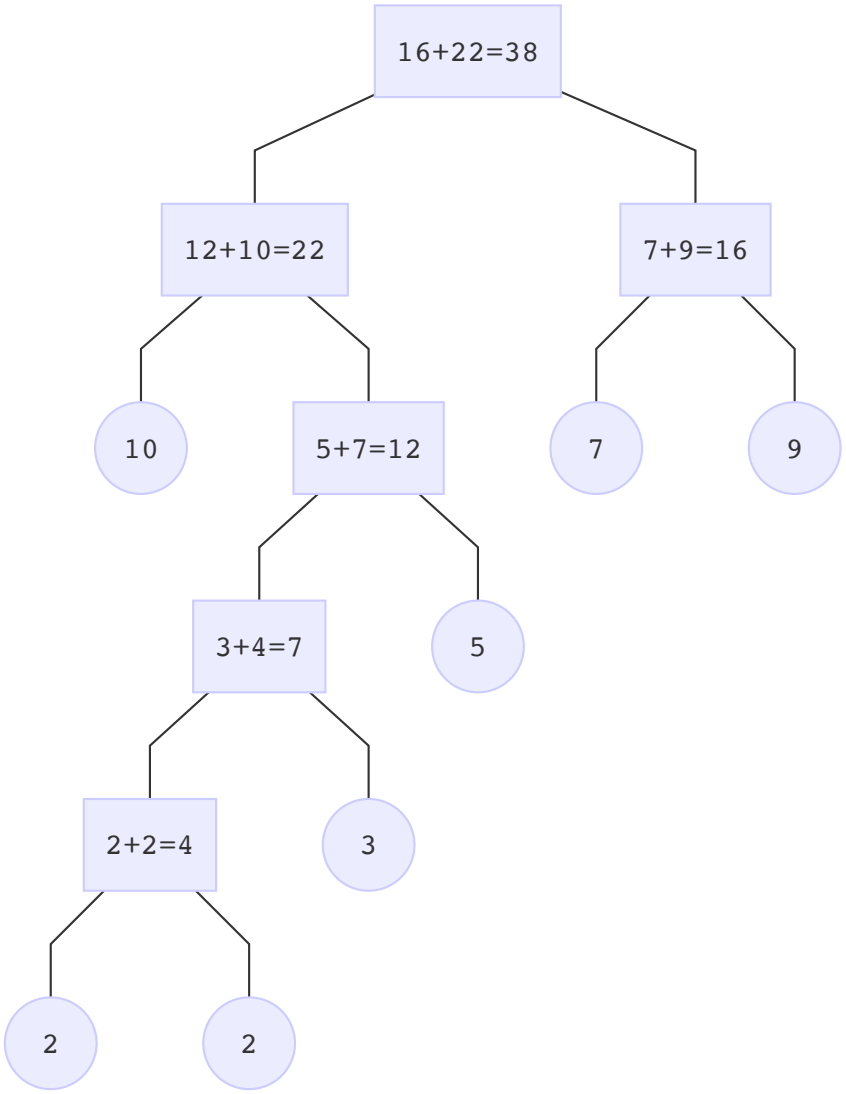


The total number of operations to merge these three files is $(m_2 + m_3) + (m_1 + m_2 + m_3)$. Notice that m_2 and m_3 contribute to the total twice as much m_1, because they were merged earlier in that process. With that in mind, to minimize the total number of operations, we need to perform the "cheapest" (fewest operations) merge at each stage.

For this algorithm, we will store the files in a priority queue where the smallest files have the highest priority. We dequeue and merge the smallest two files and place the result back in the priority queue. In this way, we will always be performing the merge with the fewest possible operations at each stage, and the "repeated" files are selected to minimize the total number of operations.

Example

File lengths: 10,7,3,5,9,2,3,2



This algorithm is identical to the Huffman code algorithm. We are minimizing the weighted path length of the merge tree with respect to the length of the files, instead of the frequencies of words/characters.