University of Nevada, Las Vegas Computer Science 477/677 Fall 2023

Answers to Assignment 7: Due Sunday November 19, 2023

1. Fill in the blanks.

   (a) In **cuckoo** hashing, no two data have the same hash value.

   (b) Using **Huffman** coding, the codons for the different symbols of a message may be written consecutively without spaces.

2. Give the asymptotic time complexity in terms of n, using Θ, O, or Ω, whichever is most appropriate.

   (a) $F(n) \geq F(n - \sqrt{n}) + n^2$

   $F(n) - F(n - \sqrt{n}) \geq n^2$

   $$\frac{F(n) - F(n - \sqrt{n})}{\sqrt{n}} \geq n^{3/2}$$

   $F'(n) = \Omega(n^{3/2})$

   $F(n) = \Omega(n^{5/2})$

   (b) $H(n) < H(n/3) + H(n/4) + 2H(n/5) + n$

   $\frac{1}{3} + \frac{1}{4} + 2 \cdot \frac{1}{5} = \frac{47}{60} < 1$

   Therefore

   $H(n) = O(n)$

   (c) $G(n) = 3(G(2n/3) + 3G(n/3)) + 5n^2$

   Thus $\gamma = 2$. $3\left(\frac{2}{3}\right)^2 + 3\left(\frac{1}{3}\right)^2 > 1$

   Through trial and error, we discover that $p = 3$:
   $3\left(\frac{2}{3}\right)^3 + 3\left(\frac{1}{3}\right)^3 = 1$

   Therefore $G(n) = \Theta(n^3)$

   (d) $F(n) = 2F(n - 1) + 1$

   Substitute $m = 2^n$, $F(n) = G(m) = G(2^n)$
   $F(n - 1) = G(2^{n-1}) = G(m/2)$
   $G(m) = 2G(m/2) + 1$
   $G(m) = \Theta(m)$ by the master theorem
   $F(n) = G(m) = \Theta(m) = \Theta(2^n)$
   $F(n) = \Theta(2^n)$

3. For each of these recursive subprograms, write a recurrence for the time complexity, then solve that recurrence.

   (a)
```
void george(int n)
 {
  if(n > 0)
   {
    for(int i = 0; i < n; i++) cout << "hello" << endl;
    george(n/2); george(n/3); george(n/6);
   }
 }
```

   Let $T(n)$ be the time to compute george(n). Then $T(n) = T(n/2) + T(n/3) + T(n/6) + n$ Thus $\gamma = 1$, and $(1/2)^\gamma + (1/3)^\gamma + (1/6)^\gamma = 1$. Thus $T(n) = \Theta(n \log n)$.

   (b)
```
void martha(int n)
 {
  if (n > 1)
   {
    martha(n-1);
    martha(n-2);
   }
  else cout << "hello world.";
 }
```

   Hint: Look at problem 0.3 on page 9 of your textbook.

   Let $T(n)$ be the time needed to compute martha(n). Our recurrence is then $T(n) = T(n-1)+T(n-2)+1$, which closely resembles the recurrence for the Fibonacci numbers, namely $F_n = F_{n-1}+F_{n-2}$. Asymptotically, the "+1" can be neglected. Now the Fibonacci numbers are approximately powers of the golden ratio ϕ, which is the solution to the quadratic equation $\phi^2 = \phi + 1$. Thus $T(n) = \Theta(F_n) = \Theta((1+\sqrt{5})/2)^n)$.

4. A hash table with of size $m = 200$ is used to store 400 data items, using a pseudo-random hash function. The average number of items in a cell ("bucket") is clearly $\frac{400}{200} = 2$, but there could be empty cells. What is the expected number of empty cells? Approximate to the nearest integer. Hint: You may need to look in a statistics textbook, or on the internet, to figure this out.

   The expected number of cells with exactly k item is $\dfrac{m\left(\frac{n}{m}\right)^k}{k!e^{\frac{n}{m}}}$. Since $m = 200$, $n = 400$ and $k = 0$, the expected number of empty cells is $\dfrac{200}{e^2} \approx 27$

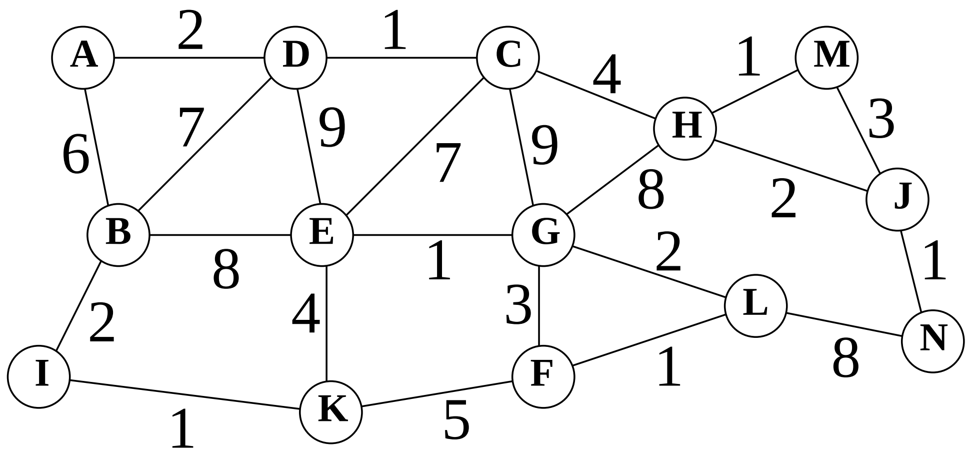
5. Let $\sigma = x_1, x_2, \ldots x_n$ be a sequence of numbers with both positive and negative terms. Write an $O(n)$ time dynamic program which finds maximum sum of any contiguous subsequence of σ. For example, if the sequence is $-1, 4, -3, 2, 7, -5, 3, 4, -6, 5, -1$ then the answer is $4 - 3 + 2 + 7 - 5 + 3 + 4 = 12$.

   Let $X[i]$ be the maximum sum of any subsequence of the first i terms, and let $Y[i]$ be the maximum sum of any subsequence which ends at x_i.

```
    $X[0] = 0$
    $Y[1] = x_1$
    $X[1] = \max(0, x_1)$
    for $i$ from 2 to $n$
        $Y[i] = \max(0, Y[i-1]) + x_i$
        $X[i] = \max(X[i-1], Y[i])$
```

   The solution is $X[n]$

6. Walk through Kruskal's algorithm to find the minimum spanning tree of the weighted graph shown below. Show the evolution of the union/find structure at several intermediate steps. Whenever there is choice between two edges of equal weight, choose the edge which has the alphabetically largest vertex. Whenever there is a union of two trees of equal weight, choose the alphabetically larger root to be the root of the combined tree. Indicate path compression when it occurs.



   We process the edges in increasing order of weight, using the given tie-breaker. The order of edges is DC, GE, KI, LF, MH, NJ, DA, IB, JH, LG, GF, MJ, HC, KE, KF, BA, ...
   This is not a complete list of edges, but once those edges are processed, the algorithm halts.

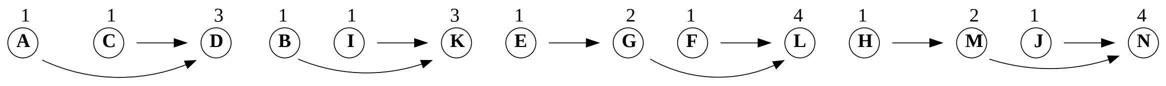
   This is the initial union/find forest.

$$\overset{1}{A}\quad \overset{1}{C}\quad \overset{1}{D}\quad \overset{1}{B}\quad \overset{1}{I}\quad \overset{1}{K}\quad \overset{1}{E}\quad \overset{1}{G}\quad \overset{1}{F}\quad \overset{1}{L}\quad \overset{1}{H}\quad \overset{1}{M}\quad \overset{1}{J}\quad \overset{1}{N}$$

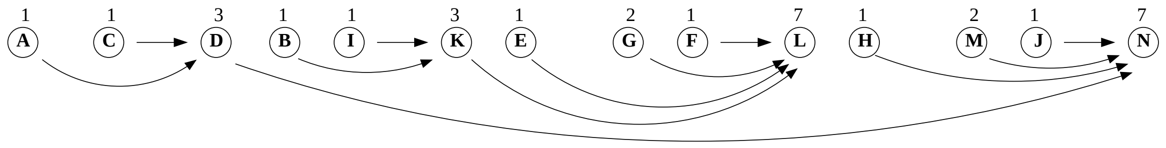
   We now process edges of length 1, namely DC, GE, KI, LF, MH, and NJ.

$$\begin{tikzcd}
\overset{1}{A} & \overset{1}{C} \arrow[r] & \overset{2}{D} & \overset{1}{B} & \overset{1}{I} \arrow[r] & \overset{2}{K} & \overset{1}{E} \arrow[r] & \overset{2}{G} & \overset{1}{F} \arrow[r] & \overset{2}{L} & \overset{1}{H} \arrow[r] & \overset{2}{M} & \overset{1}{J} \arrow[r] & \overset{2}{N}
\end{tikzcd}$$

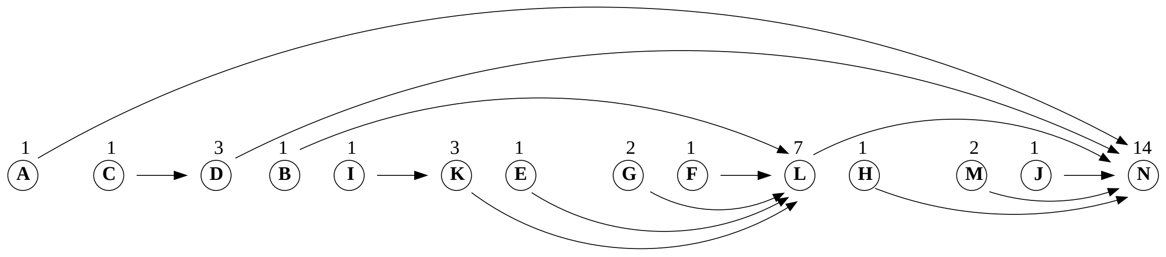
We now process edges of length 2, namely DA, IB, JH, and LG.



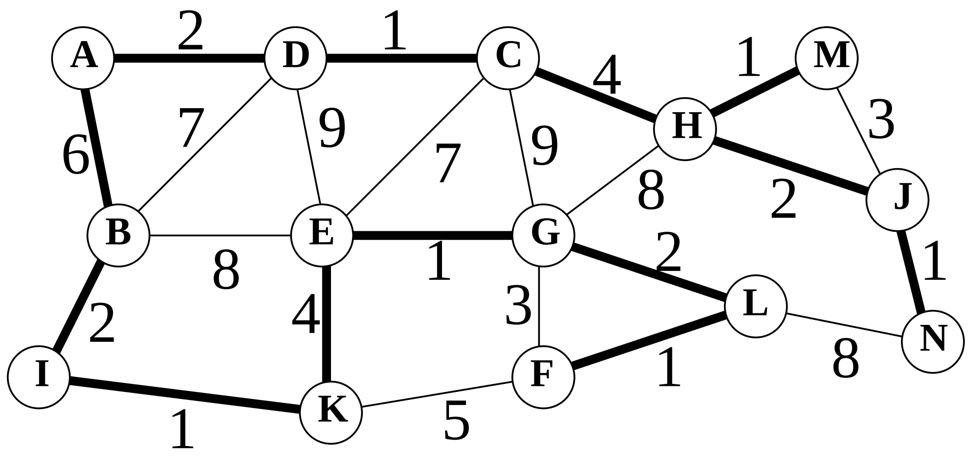
We now process edges of length 3, namely GF and MJ. There is no change in the data structure. We then process edges of length 4, namely HC and KE. There are two instances of path compression.



Finally, we process edge BA. There is one instance of path compression.

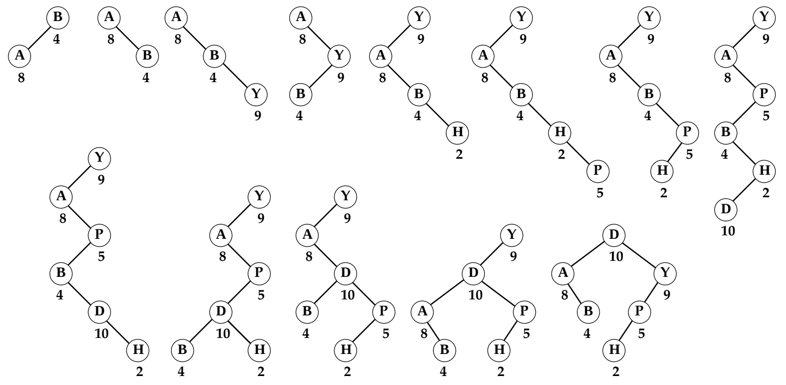


Since $n - 1 = 13$ edges have been selected, The minimal spanning tree is complete.



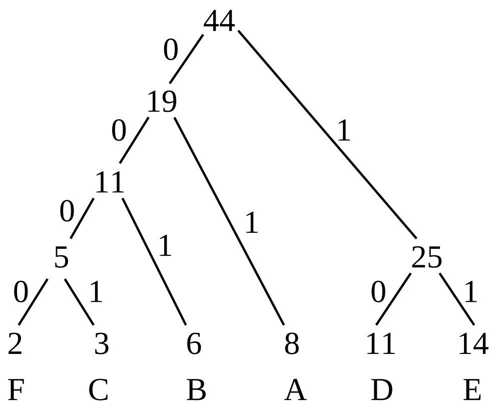
7. Insert the letters B, A, Y, H, P, D into an empty treap, where the "random" keys are given in the following table. Show the treap after each insertion and indicate all rotations.

A	8
B	4
D	10
H	2
P	5
Y	9



8. Find an optimal Huffman code on the alphabet A,B,C,D,E,F where frequencies are given in the following table.

A	8	01
B	6	001
C	3	0001
D	11	10
E	14	11
F	2	0000



9. A 3-dimensional $9 \times 7 \times 10$ rectangular array A is stored in main memory in row major order, and its base address is 8192. Each item of A takes one word of main memory, that is, one addressed location. Find the address, in main memory, of A[4][5][2].

The offset is the number of predecessors, which is $4 \times 7 \times 10 + 5 \times 10 + 2 = 332$. The address is $8192 + 332 = 8524$.

10. You are trying to construct a cuckoo hash table of size 8, where each of the 7 names listed below has the two possible hash values indicated in the array. Using Hall's marriage theorem, prove that you will fail to construct the table.

	h1	h2
Ann	1	0
Bob	4	2
Cal	0	7
Dan	1	6
Eve	1	0
Fay	6	2
Gus	5	3

The "boys" are the items in the left column and the "girls" are hash table indices. The "girls" that each "boy" "knows" are the two indices in the h1 and h2 columns. We need to assign a girl to each boy, and no girl can be assigned to more than one boy. According the Hall's marriage theorem, such an assignment exists if and only if for any set B of k boys, the set of girls known to the boys in B has cardinality at least k.
Consider the set $B = \{\text{Ann,Bob,Dan,Eve,Fay,Hal}\}$, which has cardinality 6. The set of indices that can be assigned to members of B is $\{0, 1, 2, 4, 6\}$, which has cardinality 5. Hence no assignment exists.

11. Explain how to implement a very sparse one-dimensional array using a search structure.

   Let $A[N]$ be the virtual array, where N is large and, at any given time, most of the entries of A are zero, or some other default value. We set up a search structure which stores memos. Each memo is an ordered pair (i, x) such that $A[i] = x$. If, for some i, there is no such ordered pair, then $A[i] = 0$. Fetch and store for A are implemented using find and insert for the search structure.

Arithmetic with Complex Numbers

All the arithmetic operations on complex numbers expressed in $a + b\mathbf{i}$ (Cartesion) form are computed using high school algebra, plus just one additional rule: $\mathbf{i}^2 = -1$.

Let $z = a + b\mathbf{i}$ and $w = c + d\mathbf{i}$, where a, b, c, d are real numbers.

(a) Addition: $z + w = (a + c) + (b + d)\mathbf{i}$.

(b) Subtraction: $z - w = (a - c) + (b - d)\mathbf{i}$.

(c) Multiplication: $z * w$, or simply zw, is $(ab - cd) + (ad + bc)\mathbf{i}$.

(d) Conjugate: $\overline{z} = a - b\mathbf{i}$.

(e) Modulus: $|z| = \sqrt{a^2 + b^2}$, also called the *absolute value* of z. Note that $z\overline{z} = |z|^2$.

(f) Division: $z/w = \dfrac{z}{w} = \dfrac{z\overline{w}}{w\overline{w}} = \dfrac{(a + b\mathbf{i})(c - d\mathbf{i})}{(c + d\mathbf{i})(c - d\mathbf{i})} = \dfrac{(ac + bd) + (-ad + bc)\mathbf{i}}{c^2 + d^2}$

(g) Square Root: $$\sqrt{z} = \pm\left\{\begin{array}{l} \sqrt{\dfrac{\sqrt{a^2 + b^2} + a}{2}} + \sqrt{\dfrac{\sqrt{a^2 + b^2} - a}{2}}\,\mathbf{i} \text{ if } b > 0 \\ \\ \sqrt{\dfrac{\sqrt{a^2 + b^2} + a}{2}} - \sqrt{\dfrac{\sqrt{a^2 + b^2} - a}{2}}\,\mathbf{i} \text{ if } b < 0 \end{array}\right\}$$

Note that the square root sign, $\sqrt{\ }$ indicates the positive square root of a positive real number.

12. Perform these calculations using complex numbers.

(a) What is the modulus of $\dfrac{-11+2\,\mathbf{i}}{5}$?

$\sqrt{5}$

(b) Write $\dfrac{-4+3\,\mathbf{i}}{2+\mathbf{i}}$ in $a+b\,\mathbf{i}$ form.

$-1+2\,\mathbf{i}$

(c) Write the 8th roots of unity in $a+b\,\mathbf{i}$ form.

$\dfrac{1+\,\mathbf{i}}{\sqrt{2}}$

$\mathbf{i}$

$\dfrac{-1+\,\mathbf{i}}{\sqrt{2}}$

-1

$\dfrac{-1-\,\mathbf{i}}{\sqrt{2}}$

$-\mathbf{i}$

$\dfrac{1-\,\mathbf{i}}{\sqrt{2}}$

1

(d) Write the square roots of $-1+\mathbf{i}$ in $a+b\,\mathbf{i}$ form.

$$\pm\left(\sqrt{\frac{\sqrt{2}-1}{2}}+\sqrt{\frac{\sqrt{2}+1}{2}}\right)$$

(e) Write the principle 9th root of unity in polar notation.

$\cos 40^\circ + \mathbf{i}\sin 40^\circ$

or

$\cos\dfrac{2\pi}{9} + \mathbf{i}\sin\dfrac{2\pi}{9}$

These cannot be written using square roots, as proved by Karl Friedrich Gauss, at age 19.

13. In this problem, assume that each arithmetic operation takes constant time, and that memos can be stored or fetched in constant time.

The function F is computed by the following dynamic program.

```
F(0) = 0;
for i from 1 to n
  F(i) = F(i/2) + F((i-1)/2) + i*i
```

(a) What is the asymptotic complexity of $F(n)$?

The asymptotic recurrence is: $F(n) = 2F(n/2) + n^2$. Thus $F(n) = \Theta(n^2)$.

(b) What is the asymptotic time complexity of the above dynamic program?

It takes $O(1)$ time to compute each $F(i)$, thus it takes $\Theta(n)$ time to compute $F(n)$.

(c) Write a recursive program which computes $F(n)$. What is its time complexity?

```
int F(int n)
  if(n = 0) return 0
  else return F(n/2)+F((n-1)/2)+n*n
```

in Let $T(n)$ be the asymptotic time complexity. Then $T(n) = 2T(n/2) + 1$, hence $T(n) = \Theta(n)$.

(d) What is the time complexity of computing $F(n)$ using memoization?

There are approximately $2\log_2 n$ memos needed, thus the time complexity is $\Theta(\log n)$.