CS 477/677 Answers to Study Guide for Examination November 20, 2024

1. Hashing

   (a) What is closed hashing?

   Each datum must be placed into the hash table.

   (b) What is open hashing?

   A datum can be placed in an auxiliary structure, such as a list.

   (c) What is open addressing?

   In closed hashing, a datum can be written into a cell whose index is not its hash value.

   (d) What is a perfect hash function?

   A hash function such that no two data have the same hash value.

   (e) What are the important properties of a good hash function?

   1. Deterministic.
   2. Fast to Compute.
   3. Appears random, in particular, very similar data have unrelated hash values.

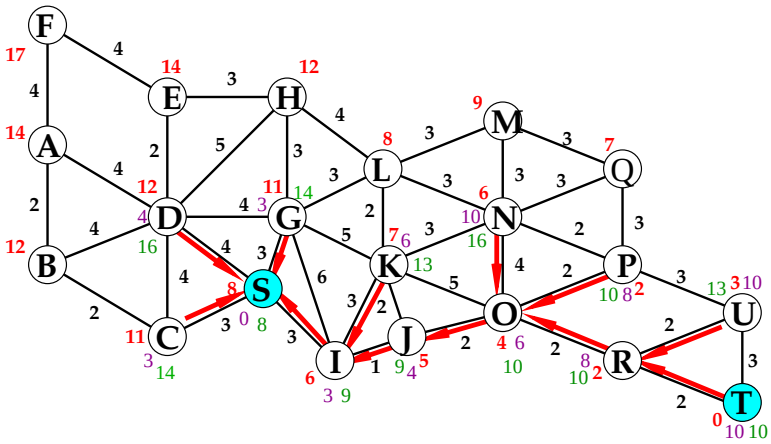
   (f) What is cuckoo hashing?

   In cuckoo hashing, each datum has more than one place it can be written in a closed hash table. If there is a collision, one datum is ejected and seeks a different location.

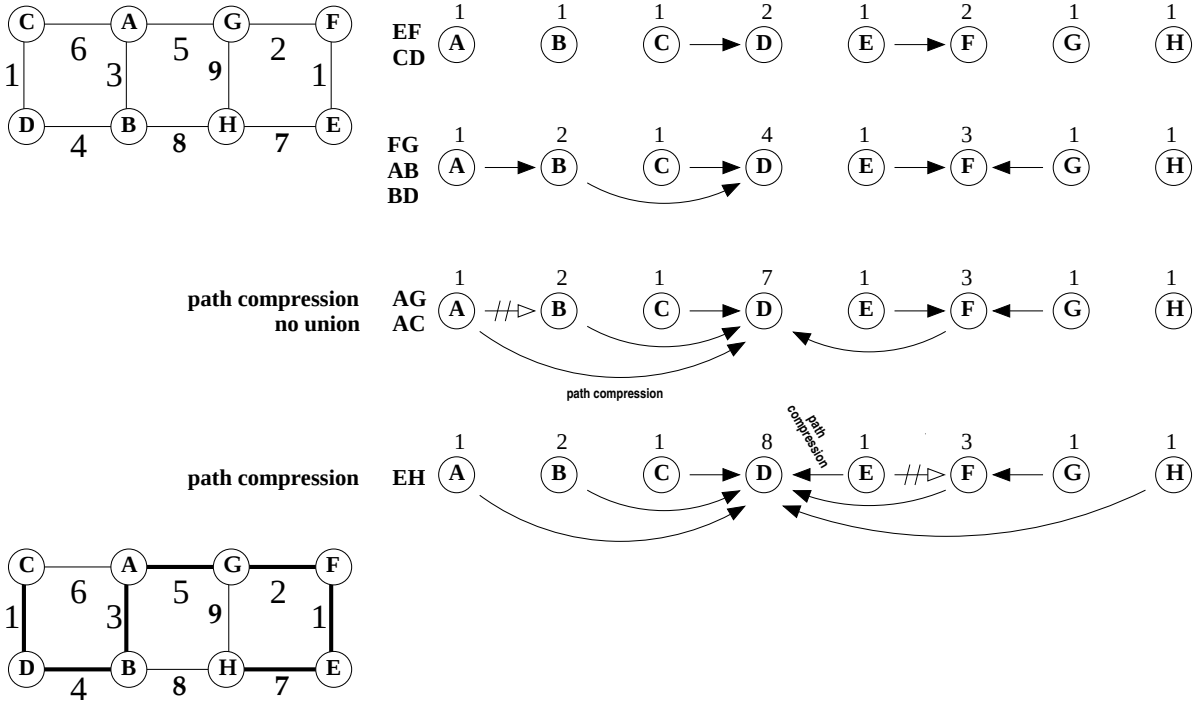
   (g) Compute the indicated mod 2 matrix product.

$$\begin{matrix} 10111100 \\ 11010110 \\ 00001011 \\ 00011110 \\ 00111010 \end{matrix} \times \begin{matrix} 111 \\ 101 \\ 001 \\ 010 \\ 100 \\ 100 \\ 011 \\ 101 \end{matrix} = \begin{matrix} 100 \\ 111 \\ 010 \\ 001 \\ 100 \end{matrix}$$

2. Work the A* algorithm for the following weighted diagraph, where the heuristic values are given in red.



3. 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. 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.



4. Fill in the blanks.

   (a) Name two greedy algorithms introduced in class this semester. **Huffman's Kruskal's**.

   (b) In closed hashing, collisions are resolved by the use of **probe** sequences.

   perfect hashing does not have collisions.

   (c) In closed hashing, if a collision occurs, a **probe sequence** can be used to locate an unused position in the hash table.

   (d) In a **cuckoo** hash table, each item has two or more possible locations, and must be stored in one of those.

   (e) (3) Which of the following three statements is closest to the truth?
   (1) In SHA256 hashing, collisions are impossible.
   (2) In SHA256 hashing, collisions occur only a few times a year in practice.
   (3) In SHA256 hashing, collisions are so unlikely that industry experts claim they never occur.

   (f) The worst case time complexity of quicksort on a list of length n.
   $O(n^2)$

   (g) The average case time complexity of quicksort on a list of length n, if pivots are chosen at random.
   $\Theta(n \log n)$

(h) The worst case time complexity of building a treap with n items.

$O(n^2)$

(i) The average case time complexity of building a treap with n items.

$\Theta(n \log n)$

(j) In an open hash table the items at each index of the table are typically shown as a linked list. However, that structure is only efficient if each list is of moderate size. In general, we use a **search structure** at each table index.

Pick one of these answers:
heap
stack
search structure

(k) A directed graph is defined to be **strongly connected** if, given any two vertices x and y, the graph contains a path from x to y.

5. A 3-dimensional $10 \times 20 \times 12$ rectangular array A is stored in main memory in column major order, and its base address is 1024. Each item of A takes two words of main memory, that is, two addressed location. Find the address, in main memory, of A[5][13][7].

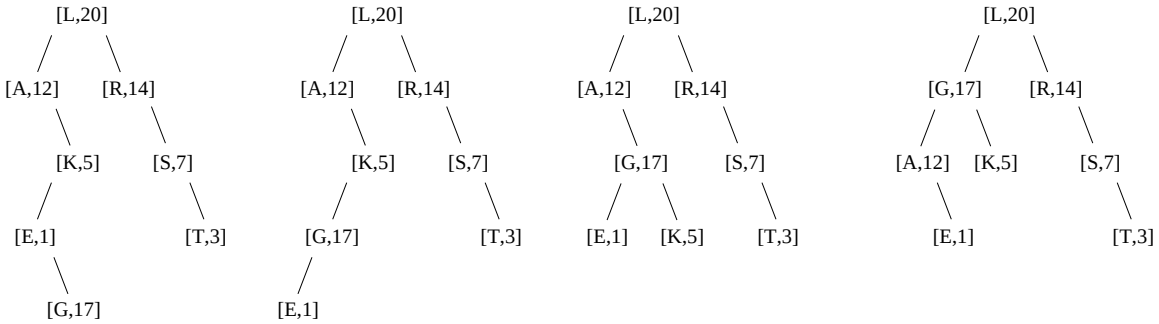
$1024 + 2 * (7 * 20 * 10 + 13 * 10 + 5) = 4094$

6. You are trying to construct a cuckoo hash table of size 8 holding 8 names. Each of the names listed below has the two possible hash values indicated in the array. Put the items into the table in alphabetic order, if possible. Instead of erasing ejected items, simply strike them out.

Ann	3	7
Bob	2	3
Dan	4	6
Eve	0	2
Fay	1	5
Gus	2	1
Hal	4	7
Jan	2	3

0	Eve
1	~~Fay~~ Gus
2	~~Bob~~ ~~Gus~~ Jan
3	~~Ann~~ Bob
4	~~Dan~~ Hal
5	Fay
6	Dan
7	Ann

7. The figure below shows a treap, where the data are letters and the nodes of the tree are memos, where the first component is the *key*, a letter, and the second component is a the *priority*, a random integer. Insert the letter G with priority 17. Show the steps.



8. Explain how to implement a sparse array using a search structure.

   Let A be the sparse virtual array. Let S be a search structure which contains ordered pairs of the form (i, x), where $A[i] = x$. To fetch the value of $A[i]$, search S for a pair (i, x). If that pair is found, return x, otherwise return a default value, such as 0. To store a value x for $A[i]$, search S for a pair (i, y). If that pair is found, replace y by x. That pair is not found, insert the pair (i, x) into S.

9. Consider the following two C++ subprograms.

```
int f(int n)
  {
   if(n > 0)
    return f(n/2)+f(n/3)+f(n/6)+n*n;
   else
    return 0;
  }

void computef(int n)
 {
  f[0] = 0;
  for(int i = 0; i <= n; i++)
   f[i] = f[i/2]+f[i/3]+f[1/6]+n*n;
 }
```

   (a) The first of those subprograms is a recursive function. What is the asymptotic value of the function f computed by the code?
   $f(n) = \Theta(n^2)$

   (b) What is the asymptotic time complexity of the computation of f(n) using the recursive function?
   The recurrence is $T(n) = T(n/2) + T(n/3) + T(n/6) + 1$ By using the Akra-Brazzi method, we obtain $T(n) = \Theta(n)$

   (c) The second subprogram uses dynamic programming, and stores values in an array. What is the asymptotic time complexity of that computation?
   $\Theta(n)$

(d) What is the asymtotic time complexity of a computation of f(n) using memoization? (Hint: it's a polylograthimic function of n)
$T(n) = \Theta(\log^2 n)$

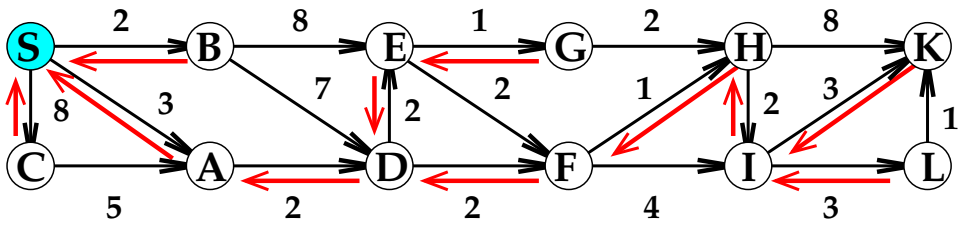
10. Write the prefix expression equivalent to the infix epression $-a * b - (-c - d) \wedge e$
(Don't forget that $\wedge$ means exponentiation.)

$-* \sim ab \wedge - \sim cde$

11. Walk through the stack algorithm to change the infix expression $-a + b \wedge c \wedge -f$ to postfix. Show the stack at each step.

stack	infile	outfile
	$-a + b \wedge c \wedge -f$	
$\sim$	$a + b \wedge c \wedge -f$	
$\sim$	$+b \wedge c \wedge -f$	a
	$+b \wedge c \wedge -f$	$a \sim$
$+$	$b \wedge c \wedge -f$	$a \sim$
$+$	$\wedge c \wedge -f$	$a \sim b$
$+\wedge$	$c \wedge -f$	$a \sim b$
$+\wedge$	$\wedge - f$	$a \sim bc$
$+\wedge\wedge$	$-f$	$a \sim bc$
$+\wedge\wedge\sim$	f	$a \sim bc$
$+\wedge\wedge\sim$		$a \sim bcf$
$+\wedge\wedge$		$a \sim bcf \sim$
$+\wedge$		$a \sim bcf \sim \wedge$
$+$		$a \sim bcf \sim \wedge\wedge$
		$a \sim bcf \sim \wedge\wedge +$

12. Walk through Dijkstra's algorithm for the following graph.

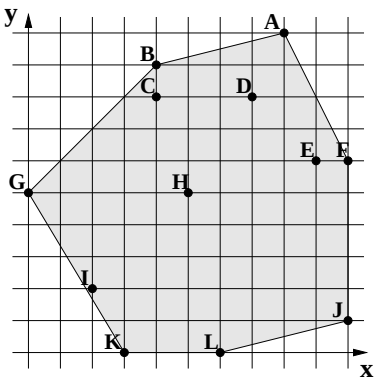


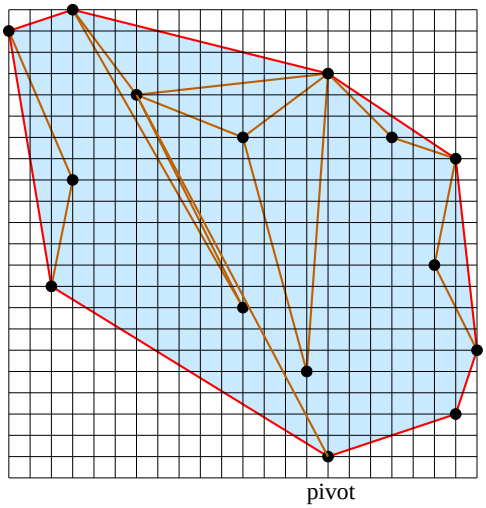
	S	A	B	C	D	E	F	G	H	I	K	L
V	0	3	2	8	~~9~~ 5	~~10~~ 7	7	8	8	~~11~~ 10	~~16~~ 13	13
back		S	S	S	~~B~~ A	~~B~~ D	D	E	F	~~F~~ H	~~H~~ I	I

13. The convex hull of a set of a finite set of points in a plane is the smallest convex polygon which encloses

the points, together with its interior. Walk through Graham Scan to find the convex hull of the points in the plane given in the second figure. (I have not gone over Graham Scan in class yet.)

Here is an example, showing the convex hull of the set {A,B,C,D,E,F,G,H,I,J,K}.

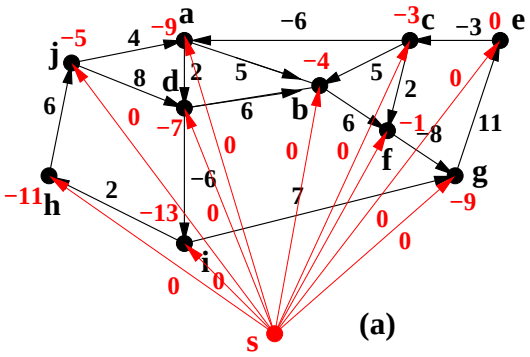


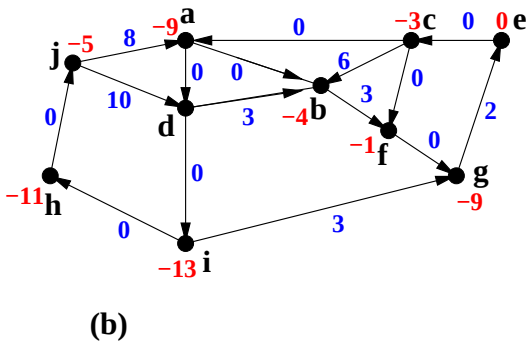


Note that the two figures refer to different sets of points in the plane.

14. Figure (a) below shows an instance of the all-pairs minpath problem. Work the first part of Johnson's algorithm on that graph, showing the adjusted weights in Figure (b).

    Do not complete the computation of Johnson's algorithm.





15. True or False.

    (a) **F** If there are 100 data items and 200 possible hash values, a collision is so unlikely that you can, in practice, assume that it won't happen.

    (b) **F** Open hashing uses open addressing.

    (c) **F** Open hashing uses probe sequences.

    (d) **F** You can avoid collisions in a hash table by making the size of the table the square of the number of data. as the data set.

    (e) **T** False overflow for a queue can be avoided by implementing the queue as a circular list.

(f) **T** If a stack is implemented as a linked list, the head of the list should be the top of the stack.

(a) **F** Kruskal's algorithm uses dynamic programming.

(b) **F** There will be no collisions if the size of a hash table is at least the square of the number of data items.

(But the expected number of collisions is less than 1.)

16. Solve each recurrence, expressing each answer using Θ.

(a) $F(n) = F(n/3) + F(2n/3) + 1$

$F(n) = \Theta(n)$

(b) $G(n) = 2G(n/4) + \sqrt{n}$

$G(n) = \Theta(\sqrt{n} \log n)$

(c) $H(n) = \log n + 1$

$H(n) = \Theta(\log^* n)$

(d) $H(n) = 4H(2n/5) + H(3n/5) + 2n^2$

$4(2/5)^2 + (3/5)^2 = 1$. Therefore $H(n) = \Theta(n^2 \log n)$.

(e) $G(n) = 4(G(n/2) + 5n^2$

$4(1/2)^2 = 1$, therefore $G(n) = \Theta(n^2 \log n)$.

(f) $F(n) = F(n - \log n) + \log^2 n$

$\frac{F(n) - F(n - \log n)}{\log n} = \frac{\log^2 n}{\log n}$

$F'(n) = \Theta(\log n)$

$F(n) = \Theta(n \log n)$

17. Find the time complexity of each of these code fragments in terms of n, using Θ notation.

(a)
```
for(int i = n; i > 1; i = i/2)
 cout < "Hello world!";
```
$\Theta(\log n)$

(b)
```
for(int i = 2; i < n; i = i*i)
 cout < "Hello world!";
```
$\Theta(\log \log n)$

(c)
```
for(int i = 1; i < n; i++)
 for(int j = i; j < n; j=2*j)
  cout < "Hello world!";
```
$\Theta(n)$

18. The asymptotic complexity of the Floyd/Warshall algorithm is $\Theta(n^3)$

19. The asymptotic complexity of Dijkstra's algorithm algorithm is $O(m \log n)$

20. Here is another coin-row problem. You have a row of coins of various values, where the value of the i^{th} coin is $V[i] > 0$. Write pseudocode which finds the maximum value of a subset of coins, where the set may not contain coins which are either adjacent or just one apart in the row. That is, if the set contains the i^{th} coin, it may not contain either the $(i+1)^{\text{st}}$ coin or the $(i+2)^{\text{nd}}$ coin. For example, if the coins are ⓐ ⓑ ⓒ ⓓ ⓔ ⓕ ⓖ ⓗ in that order, the subset may be {ⓐ, ⓓ, ⓗ}, but not {ⓑ, ⓓ, ⓖ}. We have two dynamic programs for this problem.

- Algorithm I: let $A[i]$ be the maximum sum of any legal sequence ending at i. Then the answer is $max(A[n-2], A[n-1], A[n])$ where the values of A are computed as follows.

```
A[1] = V[1]
A[2] = V[2]
A[3] = V[3]
A[4] = V[4] + A[1]
A[5] = V[5] + max(A[1],A[2])
for i from 6 to n
  A[i] = V[i] + max(A[i-5],A[i-4],A[i-3])
```

- Algorithm II: Let A[i] be the maximum sum of any legal subsequence of the first i coins. The answer is then $A[n]$ where the values of A are computed as follows.

```
A[1] = V[1]
A[2] = max(V[2],A[1])
A[3] = max(V[3],A[2])
for i from 4 to n
  A[i] = max(A[i-1],V[i] + A[i-3])
```

21. Execute heapsort for the list BXQVRST. Show the array at each step.

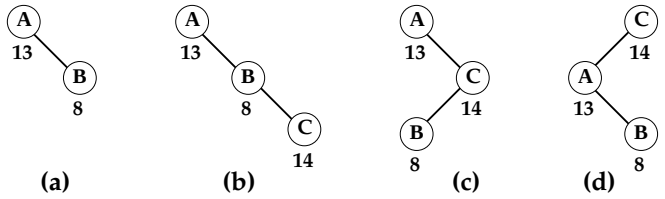
B	X	Q	V	R	S	T
B	X	T	V	R	S	Q
X	B	T	V	R	S	Q
X	V	T	B	R	S	Q
Q	V	T	B	R	S	**X**
V	Q	T	B	R	S	**X**
V	R	T	B	Q	S	**X**
S	R	T	B	Q	**V**	**X**
T	R	S	B	Q	**V**	**X**
Q	R	S	B	**T**	**V**	**X**
S	R	Q	B	**T**	**V**	**X**
B	R	Q	**S**	**T**	**V**	**X**
B	R	Q	**S**	**T**	**V**	**X**
R	B	Q	**S**	**T**	**V**	**X**
Q	B	**R**	**S**	**T**	**V**	**X**
B	**Q**	**R**	**S**	**T**	**V**	**X**
B	**Q**	**R**	**S**	**T**	**V**	**X**

22. **Huffman's** algorithm finds a binary code so that the code for one symbol is never a prefix of the code for another symbol.

23. An acyclic directed graph with 9 vertices must have at least **9** strong components. (Must be exact answer.)

24. In **open hashing** or **separate chaining** there can be any number of items at a given index of the hash table.

25. You need to store the items A, B, and C, in that order, in a treap. The priority for A is 13, for B is 8, and for C is 14. Use maxheap order. Draw the resulting treap after each insertion, and show each rotation.



We first insert A, then B. Heap order is preserved, so no rotation is necessary.

We next insert C. To restore heap order, we do a left roatation at B.

Heap order is still not restored. We do a left rotation at A. Heap order is then restored.

26. Consider the function F computed by the recursive code given below.

    (a) What is the asymptotic complexity of $F(n)$?
    Recurrence: $F(n) = 3F(n/3) + n^2$
    Solution: $F(n) = \Theta(n^2)$

    (b) What is the asymptotic time complexity of the recursive code when it computes $F(n)$?
    Recurrence: $T(n) = 3T(n/3) + 1$
    Solution: $T(n) = \Theta(n)$

    (c) What is the asymptotic time complexity of a memoization algorithm which computes $T(n)$?
    $T(n) = \Theta(\log n)$

```
int F(int n)
 {
  if(n < 3) return 1;
  else return F(n/3)+2*F((n+1)/3)+n*n;
 }
```

27. If the array $A[5][7]$ is stored in column-major order, how many predecessors does $A[3][4]$ have?

    4*5 + 3 = 23

28. You are implementing a 3D triangular array A where $A[i][j][k]$ is defined for $i \geq j \geq k \geq 0$, as a one-dimensional subarray of main memory, and you wish to store A in row-major order, with base address

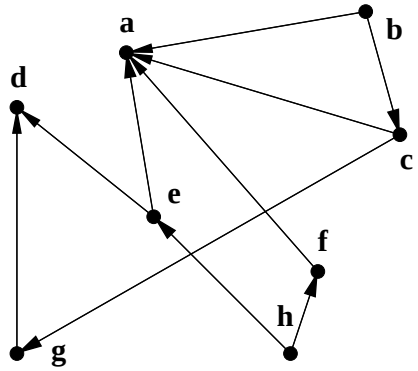
1024. Each term of A takes one place in main memory. What would be the address, in main memory, of $A[7][4][3]$?

I have not covered multidimensional triangular arrays. I will postpone that discussion until after the third examination.

Here is the answer anyway, though:

There are choose(7+2,3)+choose(4+1,2)+choose(3,1) = 84+10+3 = 97 predecessors of A[7][4][3], hence A[7][4][3] is located at address 97+1024 = 1121

29. Write the array of in-neighbor lists and the array of out-neighbor lists for the directed graph below.



a	
b	a,c
c	a,g
d	
e	a,d
f	a
g	d
h	e,f

a	b,c,e,f
b	
c	b
d	e,g
e	h
f	h
g	c
h	

30. Consider the following recursive C++ function.

```
int f(int n)
 {
  if(n > 0) return f(n/2)+f(n/4)+f(n/4 + 1)+n;
  else return 0;
 }
```

(a) What is the asymptotic complexity of f as a function of n, using Θ notation?

The recurrence is $f(n) = f(n/2) + 2f(n/4) + n$
By the Akra-Brazzi theorem, $f(n) = \Theta(n \log n)$.

(b) What is the asymptotic time complexity of this code as a function of n, using Θ notation?

The recurrence is $T(n) = T(n/2) + 2T(n/4) + 1$
By the Akra-Brazzi theorem, $T(n) = \Theta(n)$.

(c) Write pseudo-code for a dynamic programming algorithm to compute $f(n)$ for a given n. What is the asymptotic time complexity of your code as a function of n, using Θ notation?

```
f[0] = 0;
for(int i = 1; i <= n; i++)
  f[i] = f[i/2] + f[i/4] + f[i/4 + 1] + i;
cout << f[n] << endl;
```

The value of $f(i)$ is computed for each i up to n. The answer is $\Theta(n)$.

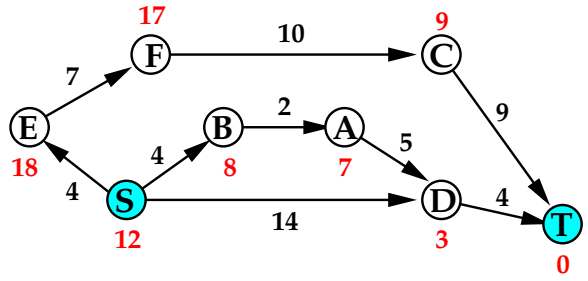
(d) Write pseudo-code for a memoization algorithm to compute $f(n)$ for a given n. What is the asymptotic complexity of the algorithm in terms of n, using Θ notation?

Represent the subproblem f[i] by the integer i. There is one subproblem for each integer from 0 to n. The subproblems are the vertices of a directed graph. There is an arc from i to j if the computation of f[j] requires the value of f[i]. We need to find the number of predecessors of n in this directed graph.

It helps to work out an example. Let n = 1785. We need to compute f for the following integers:
1785, 892, 446, 447, 223, 224, 111, 112, 55, 56, 27, 28, 29, 13, 14, 15, 6, 7, 8, 3, 4, 1, 2, 0.

Except for the smallest few, the predecessors are in blocks where each block starts with n divided by a power of 2 and has at most three members. Thus the number of predecessors is approximately $3\log_2$n. Thus the number of memos stored is $\Theta(\log n$. The search time needed is $O(\log n \log\log n)$ if the time required for a search or insert is asymptotically the logarithm of the size of the search structure. If you ignore search and insert time, the time complexity is $O(\log n)$.

31. Walk through the A^* algorithm for the weighted directed graph shown below, where the pair is (S, T). The heuristic is shown as red numerals.



.

Show the arrays and the contents of the heap at each step. h is the heuristic, g is the current distance from the source, f is the sum of h and g. Label processed vertices with g and f, and show the backpointers.

Heap: S

	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
g	0							
f	12							
back								

Heap: BDE

	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
g	0		4		14	4		
f	12		12		17	22		
back			S		S	S		

Heap: ADE

	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
g	0	6	4		14	4		
f	12	13	12		17	22		
back		B	S		S	S		

Heap: DE

	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
g	0	6	4		11	4		
f	12	13	12		14	22		
back		B	S		A	S		

Heap: TE

	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
g	0	6	4		11	4		15
f	12	13	12		14	22		15
back		B	S		A	S		D

Heap: E

	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
g	0	6	4		11	4		15
f	12	13	12		14	22		15
back		B	S		A	S		D

T is fully processed, and we are done. The shortest path from S to T is (S,B,A,D,T) obtained by following the back pointers.

32. Find the Levenshtein edit distance from the word "mennoover" to the word "maneuver." Show the matrix.

		m	a	n	e	u	v	e	r
	0	1	2	3	4	5	6	7	8
m	1	0	1	2	3	4	5	6	7
e	2	1	1	2	2	3	4	5	6
n	3	2	2	1	2	3	4	5	6
n	4	3	3	2	2	3	4	5	6
o	5	4	4	3	3	3	4	5	6
o	6	5	5	4	4	4	4	5	6
v	7	6	6	5	5	5	4	5	6
e	8	7	7	6	5	6	5	4	5
r	9	8	8	7	6	7	6	5	4

The edit distance is 4.