University of Nevada, Las Vegas Computer Science 477/677 Spring 2026

Study for Examination April 15, 2026

Unfinished as of Mon Apr 13 10:24:49 PM PDT 2026. I'll get back to it in the morning.

1. True or False.

   (a) **F** If there are 100 data items and 10000 possible hash values, collisions are so unlikely that you can, in practice, assume that they won't happen.

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

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

   (d) **T** If a stack is implemented as a linked list, the head of the linked list should hold the top item of the stack.

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

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

   (g) **T** A hash function should appear to be random, but cannot actually be random.

   (h) **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.

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

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

   (k) **F** You can avoid collisions in a hash table by making the table twice as large as the data set.

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

   (m) **T** If a stack is implemented as a linked list, the head of the linked list should hold the top item of the stack.

2. Solve each of these recurrences, giving an asymptotic answer.

   (a) $F(n) = F(\log n) + 1$
   $F(n) = \log^* n$

   (b) $F(n) = F(5n/13) + F(12n/13) + n$
   $F(n) = \Theta(n^2)$

   (c) $F(n) = F(n-1) + 2F(n-2) + 1$
   $F(n) = \Theta(2^n)$

   (d) $F(n) = 5F(2n/3) + F(n/3) + 2n^4$
   $F(n) = \Theta(n^4)$

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

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

(f) $F(n) = 4F(n/2) + n^2$

$F(n) = \Theta(n^2) \log n)$

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

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

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

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

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

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

(j) $G(n) = 4G(n/2) + n^2 + 1$

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

(k) $F(n) = 4F(n/2) + n^2$

$F(n) = O(n^2 \log n)$

(l) $G(n) = G(4n/5) + G(3n/5) + n^2$

$G(n) = \Theta(n^2 \log n)$

(m) $T(n) = T(3n/10) + T(n/5) + n$

$T(n) = \Theta(n)$

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

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

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

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

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

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

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

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

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

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

Simplify the fraction:

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

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

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

(s) Consider the following recursive C++ function. Find the asymptotic complexity of f as a function of n using Θ notation.

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

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

3. Fill in the blanks.

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

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

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

(d) Name three kinds of search structurees.

Here are three that are commonly used. **list**, **hash table binary search tree**

(e) **perfect** hashing, which can be used by a compiler to identify reserved words, does not have collisions.

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

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

(h) Which of the following three statements is closest to the truth?
(1) In SHA256 hashing, collisions are impossible.
(2) In SHA256 hashing, collisions occur no more than once a year in practice.
(3) In SHA256 hashing, collisions are so unlikely that industry experts claim they never occur.
Answer is (3).

(i) A **perfect** hash function fills the hash table exactly with no collisions.

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

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

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

(m) The asymptotic expected time to find the median item in an unordered array of size n, using a good randomized algorithm, is $\Theta(n)$

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

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

(p) The worst case time complexity of quicksort on a list of length n.

$O(n^2)$

(q) The average case time complexity of quicksort on a list of length n, if pivots are chosen at random.

$\Theta(n \log n)$

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

$O(n^2)$

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

$\Theta(n \log n)$

(t) In an open hash table of size m holding n data items, 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

(u) 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.

4. Write a recurrence for the time complexity of the code below, and solve that recurrence, giving an asymptotic function of n.

```
void george(int n)
 {
  if(n > 1)
   {
    for(int i = 0; i < n; i++) cout << "hello" << endl;
    george(n/2); george((n+1)/2);
   }
 }
```

$T(n) = T(n/2) + T((n+1)/2) + n$

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

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, where each of the 8 names listed below has the two possible hash values indicated in the array. Put the items into the table, if possible. Instead of erasing ejected items, simply cross them out, so that I can tell that you worked it properly.

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

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

7. 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 the maximum sum of any contiguous subsequence of σ. For example, if the sequence is $-1, 4, -3, 2, 7, -5, 3, 4, -8, +6$ then the answer is $4 - 3 + 2 + 7 - 5 + 3 + 4 = 12$.

   Define:
   $X[i]$ = maximum sum of any contiguous subsequence of $x_1, \ldots x_i$
   $Y[i]$ = maximum sum of any contiguous subsequence which ends at x_i

   $Y]i]$ may be negative, but we do not allow any value of $X[i]$ to be negative. If no positive value of $X[i]$ exists, we assign the default value 0.

   Now, the program:

   $Y[1] = x_1$
   $X[1] = max(0, Y[1])$
   For(int $i = 2$; $i \leq n$; $i++$)
   {
   $Y[i] = x_i + max(0, Y[i-1])$
   $X[i] = max(X[i-1], Y[i])$
   }
   return $X[n]$

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. What is the asymptotic complexity of the following code, in terms of n.

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

We are talking about the time complexity of the code, not the value of $f(n)$. The recurrence is

$T(n) = T(n/2) + T(n/3) + T(n/6) + 1$

$T(n) = \Theta(n)$.

(a) The first of those subprograms is a recursive function. What is the asymptotic value of f(n)?

$\Theta(n^2)$

(b) What is the asymptotic time complexity of the computation of f(n) using dynamic programming?
$\Theta(n)$

$\Theta(n)$

(c) What is the asymtotic time complexity of a computation of f(n) using memoization?
$\Theta(\log^2 n)$

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

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

(a) What is the asymptotic complexity of $F(n)$? $\Theta(n^2)$

(b) What is the asymptotic time complexity of the recursive code when it computes $F(n)$? $\Theta(n^{\log_3 2}$

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

11. Find the asymptotic complexity of each of these code fragments.

(a)
```
for(int i = 0; i < n; i++)
  for(int j = i; j > 0; j = j/2)
```

$\Theta(n \log n)$

(b)
```
for(int i = n; i > 1; i = i/2)
 cout < "Hello world!";
```

$\Theta(\log n)$

(c)
```
for(int i = 2; i < n; i = i*i)
 cout < "Hello world!";
```

$\Theta(\log \log n)$

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

$\Theta(n)$

12. Find the asymptotic time complexity in terms of n using Θ notation of this recursive code.

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

Don't confuse computation time with the value of F. Using the master theorem, we have $F(n) = \Theta(n^2)$.

To compute the asymtotic time complexity $T(n)$, We assume that any arithmetic operation takes constant time. The recurrence is $T(n) = 2T(n/3) + 1$. Do you see why? The code doesn't compute $F((n+1)/3)$ twice. It computes it once, then multiplies the answer by 2. Unfortunately, the answer contains an irrational exponent.

$T(n) = \Theta(n^{\log_3 2}$.

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

4*5 + 3 = 23

14. You are implementing a 3D triangular array A where $A[i][j][k]$ is defined for $0 \leq k \leq j \leq i \leq 4$, a total of 35 entries (Is that correct?), and is stored as a one-dimensional subarray of main memory 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 $A[4][2][1]$?

15. You are given an acyclic directed graph $G = (V, E)$ where each arc has a positive weight. If (x, y) is an arc, we write $w(x, y)$ for the weight of that arc. Describe a dynamic programming algorithm which calculates the directed path through G of maximum weight. (The weight of a path is defined to be the sum of the weights of its constituent arcs.) (Hint: your algorithm should use words such as "in-neighbor" or "out-neighbor.")

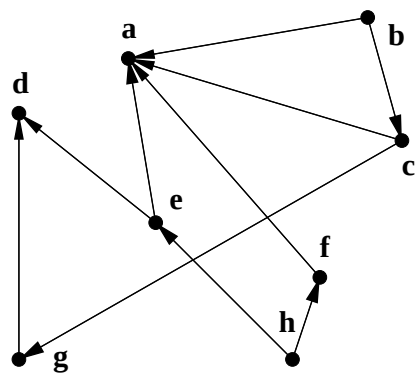
Write $W(x)$ for the largest weight of any directed path starting at x. Here is our program.

Visit all vertices in topological order. For each vertex x:

If x has no in-neighbors, let $W(x) = 0$. Otherwise let $W(x)$ be the maximum value of $W(y) + w(y, x)$ over all in-neighbors y of x.

Return the maximum value of $W(x)$ over all vertices.

16. The directed graph shown below can be stored as an array of out-neighbor lists as shown.



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	

Write the array of in-neighbor lists for the same graph.

17. 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 $f(n)$ as a function of n, using Θ notation?

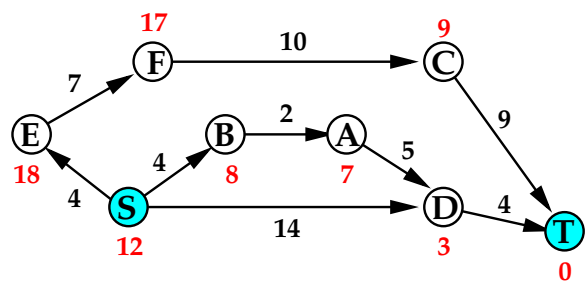
recurrence is $f(n) = f(n/2) + 2f(n/4) + n$ the Akra-Bazzi theorem, $f(n) = \Theta(n \log n)$.

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

The recurrence is $T(n) = T(n/2) + T(n/4) + 1$

By the Akra-Bazzi theorem, $T(n) = \Theta(n^{\alpha})$. where α is some hideous irrational number (Sorry, I didn't notice this.)

18. 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, f is the current distance from the source, g is the sum of h and f, while back is the backpointer.

Heap: S

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

Heap: BDE

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

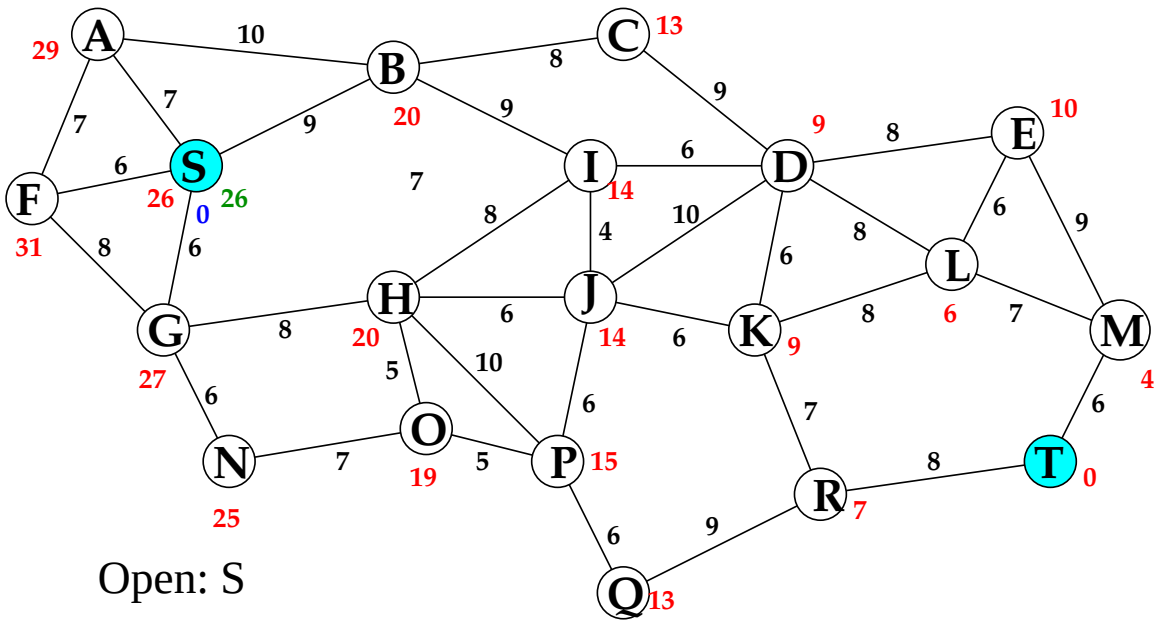
T is fully processed, and we are done. The shortest path from S to T is (S,B,A,D,T) obtained by following the back pointers.

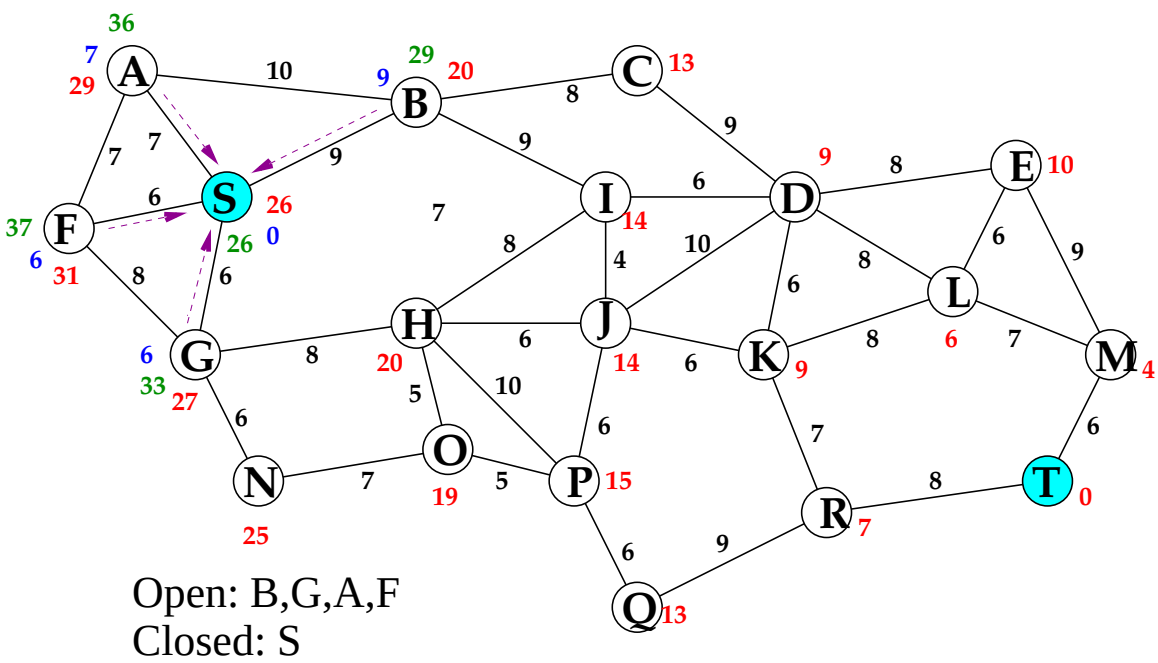
19. Find the Levenshtein edit distance from the word "mennoover" to the word "maneuver." Show the matrix.

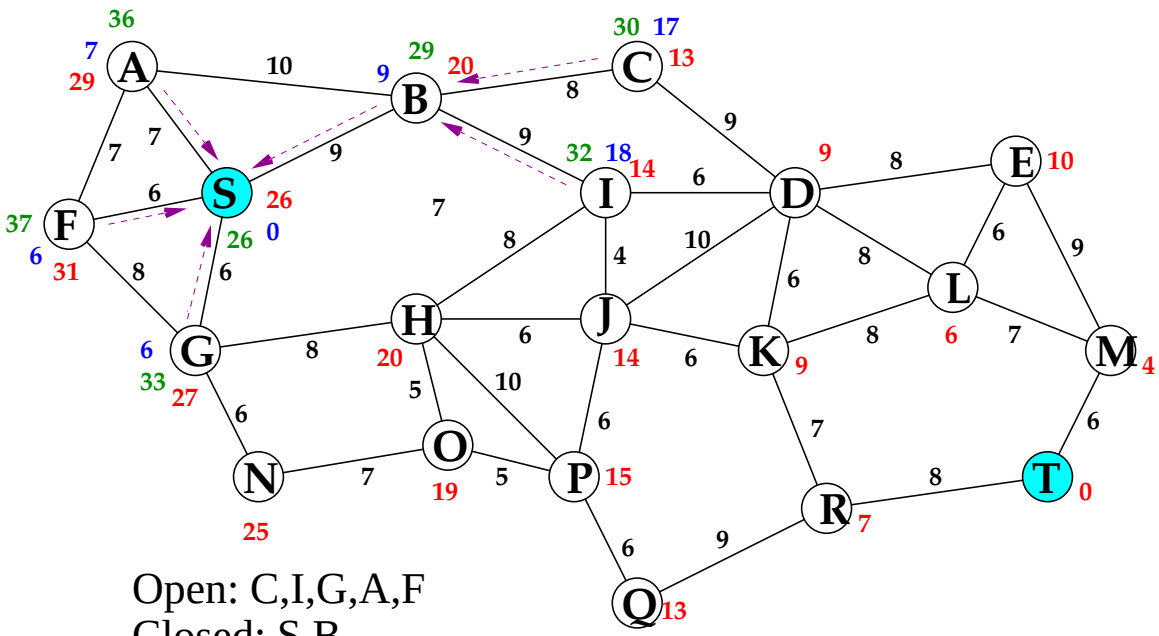
		m	a	n	e	u	v	e	r
	0	1	2	3	4	5	6	7	8
m	1	0	1	2	3	4	5	6	7
e	2	1	1	2	2	3	4	5	6
n	3	2	2	1	2	3	4	5	6
n	4	3	3	2	2	3	4	5	6
o	5	4	4	3	3	3	4	5	6
o	6	5	5	4	4	4	4	5	6
v	7	6	6	5	5	5	4	5	6
e	8	7	7	6	5	6	5	4	5
r	9	8	8	7	6	7	6	5	4

The edit distance is 4.

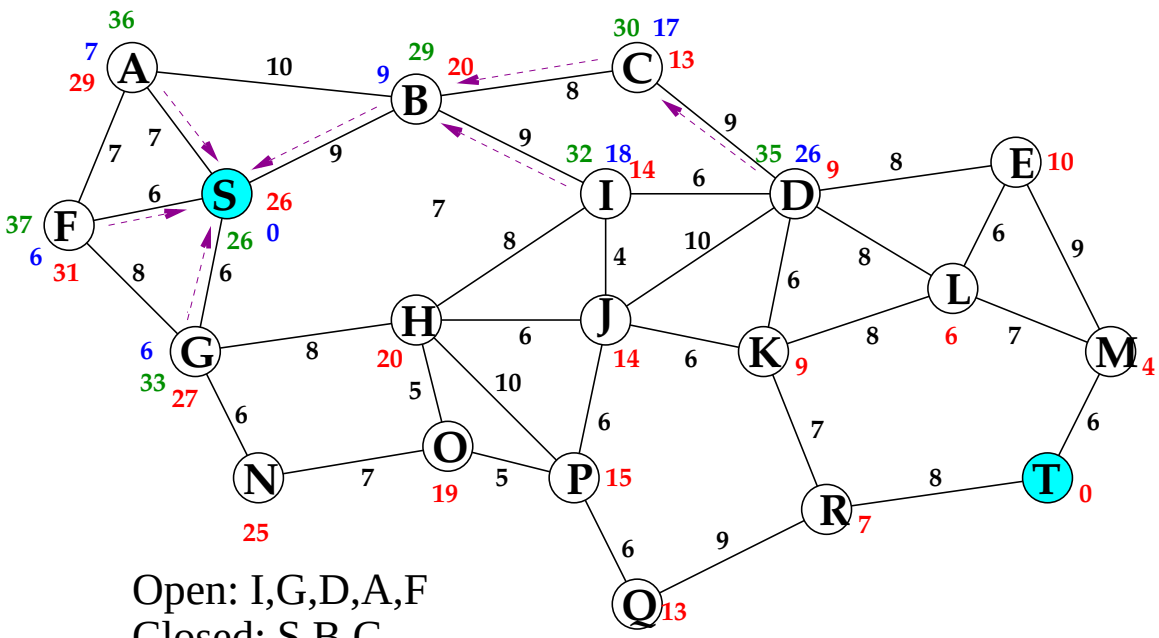
20. Walk through the A^* algorithm to find the least cost path from S to T. The values of the heuristic function are indicated in red.



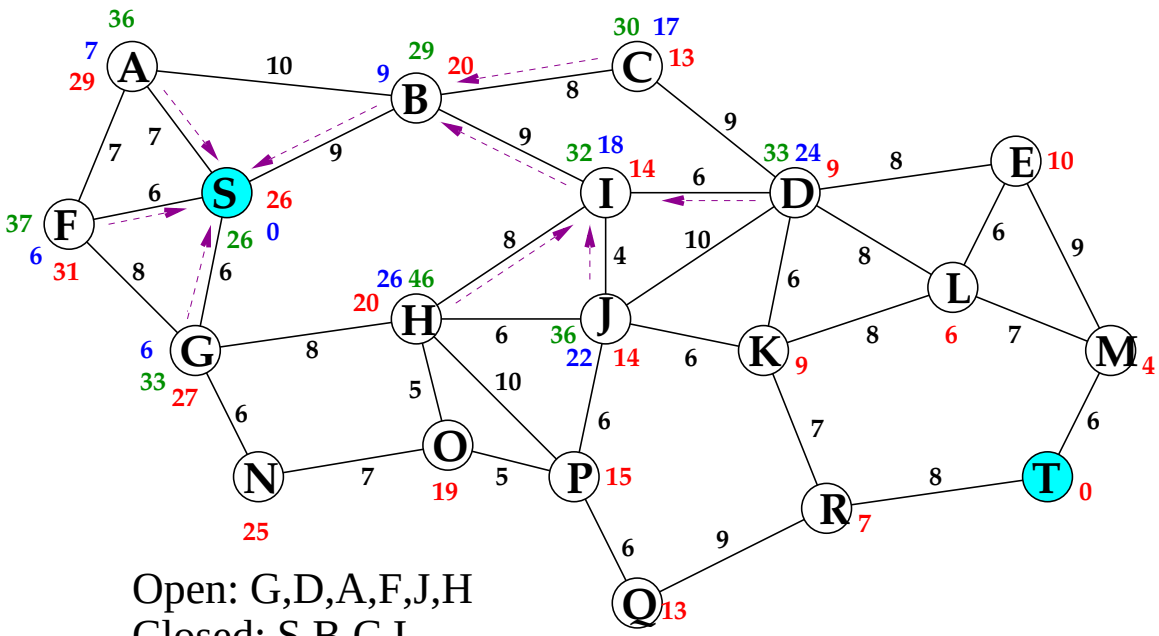




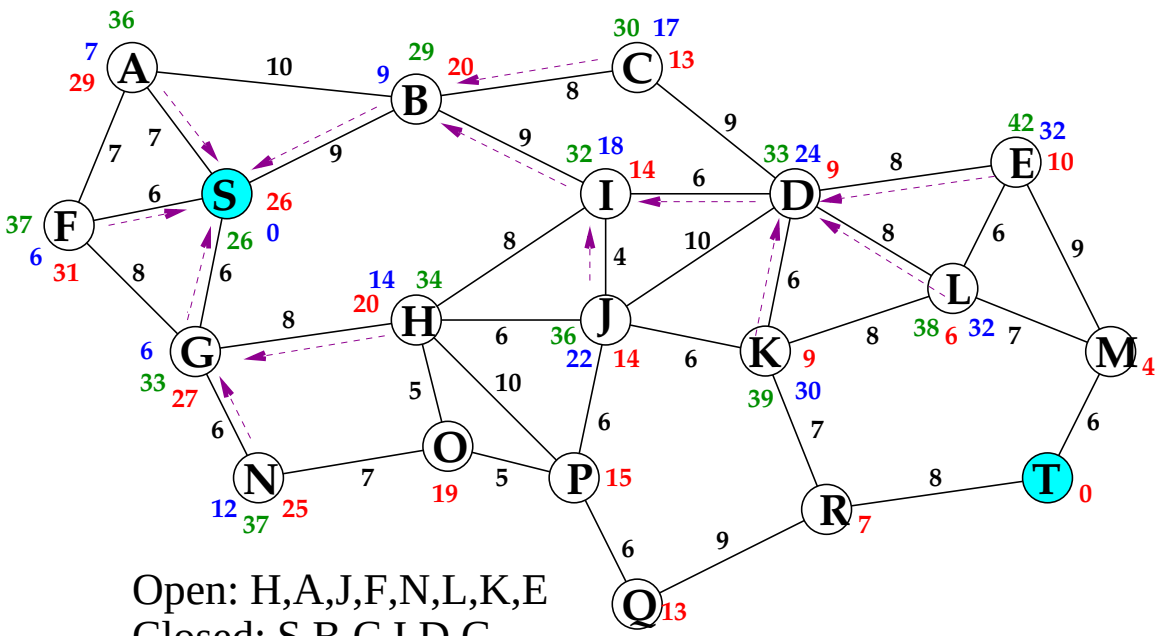
Open: C,I,G,A,F
Closed: S,B



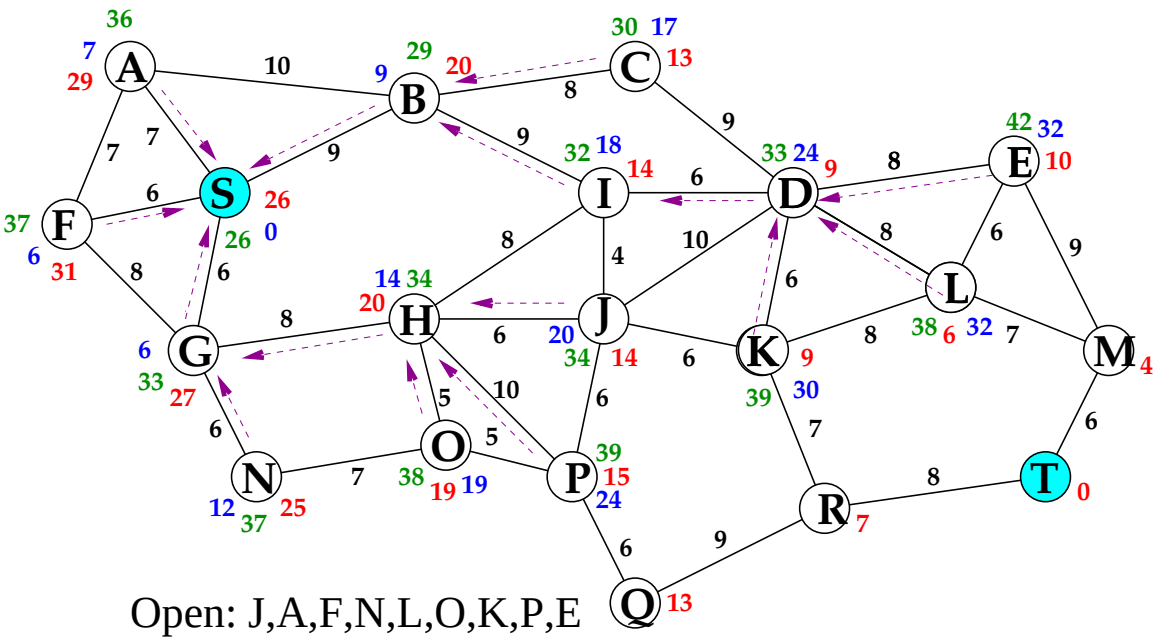
Open: I,G,D,A,F
Closed: S,B,C



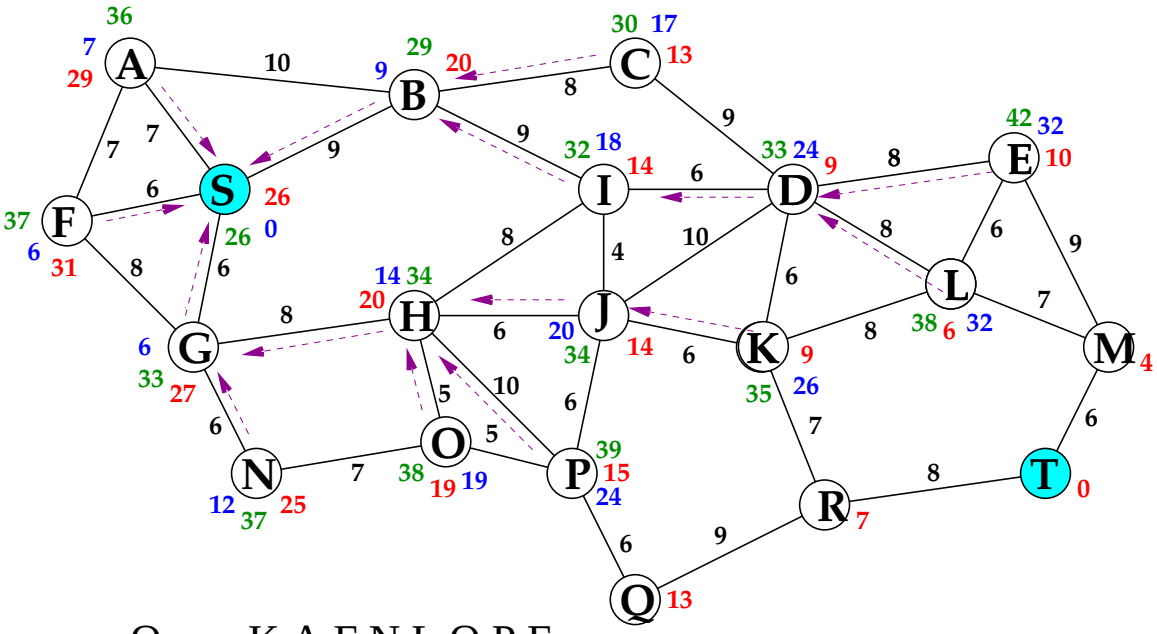
Open: G,D,A,F,J,H
Closed: S,B,C,I



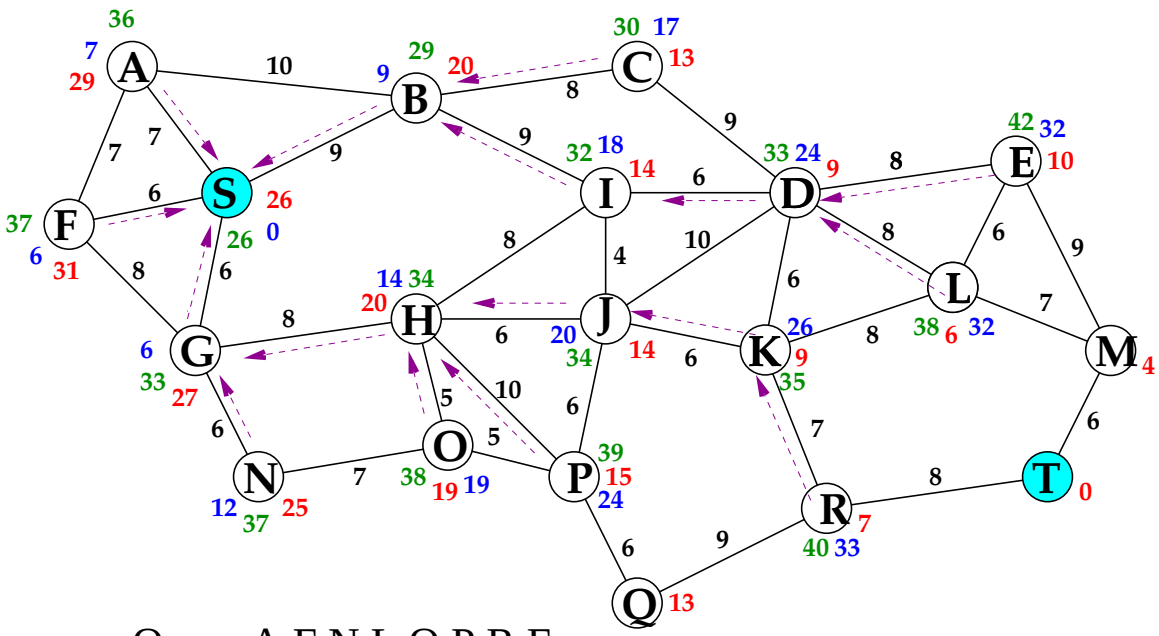
Open: H,A,J,F,N,L,K,E
Closed: S,B,C,I,D,G



Open: J,A,F,N,L,O,K,P,E
Closed: S,B,C,I,D,G,H

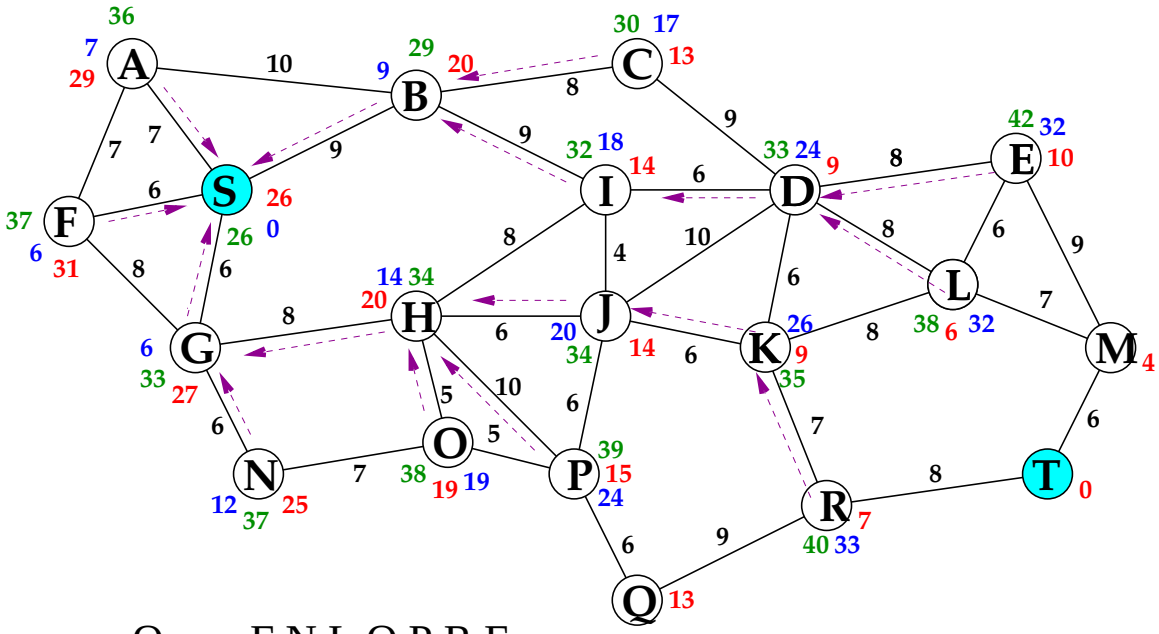


Open: K,A,F,N,L,O,P,E
Closed: S,B,C,I,D,G,H,J



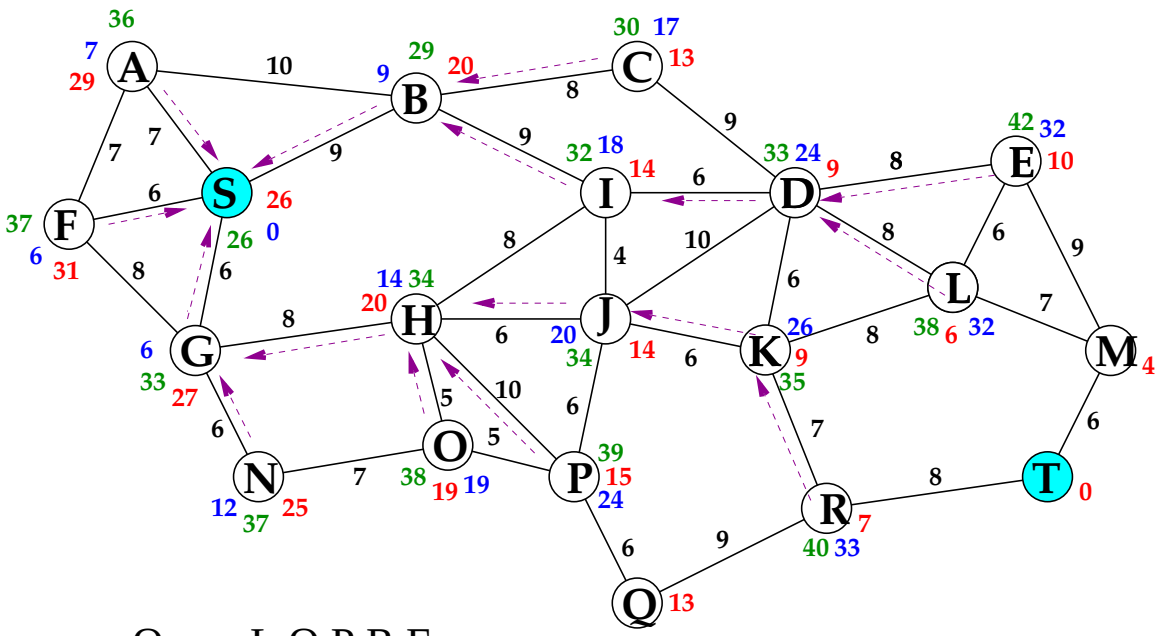
Open: A,F,N,L,O,P,R,E

Closed: S,B,C,I,D,G,H,J,K



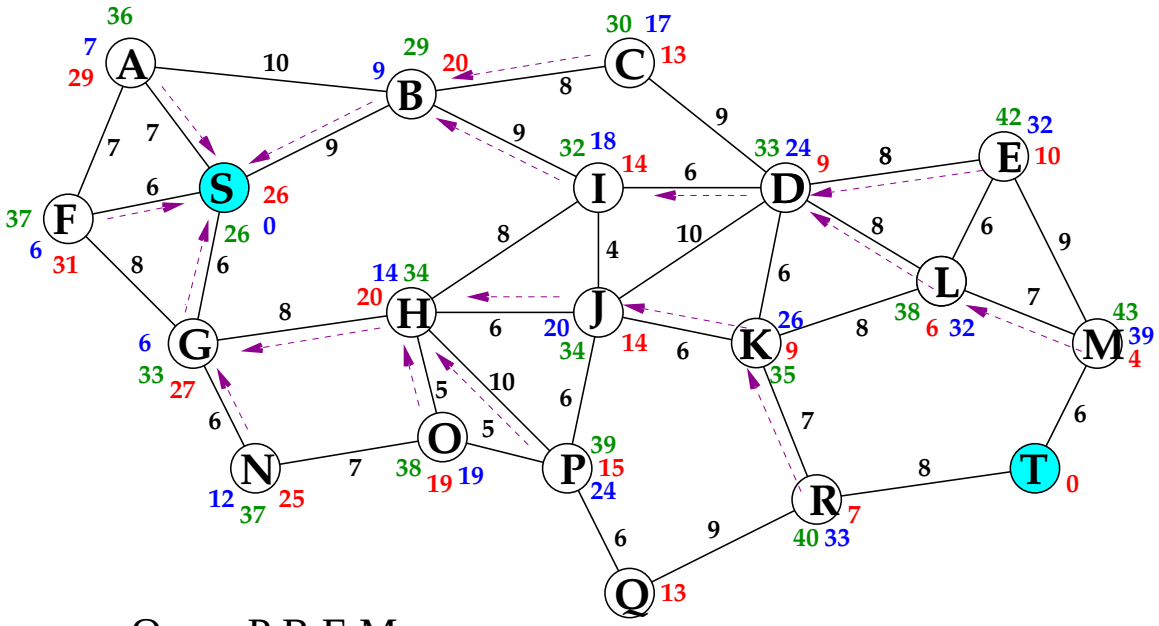
Open: F,N,L,O,P,R,E

Closed: S,B,C,I,D,G,H,J,K,A



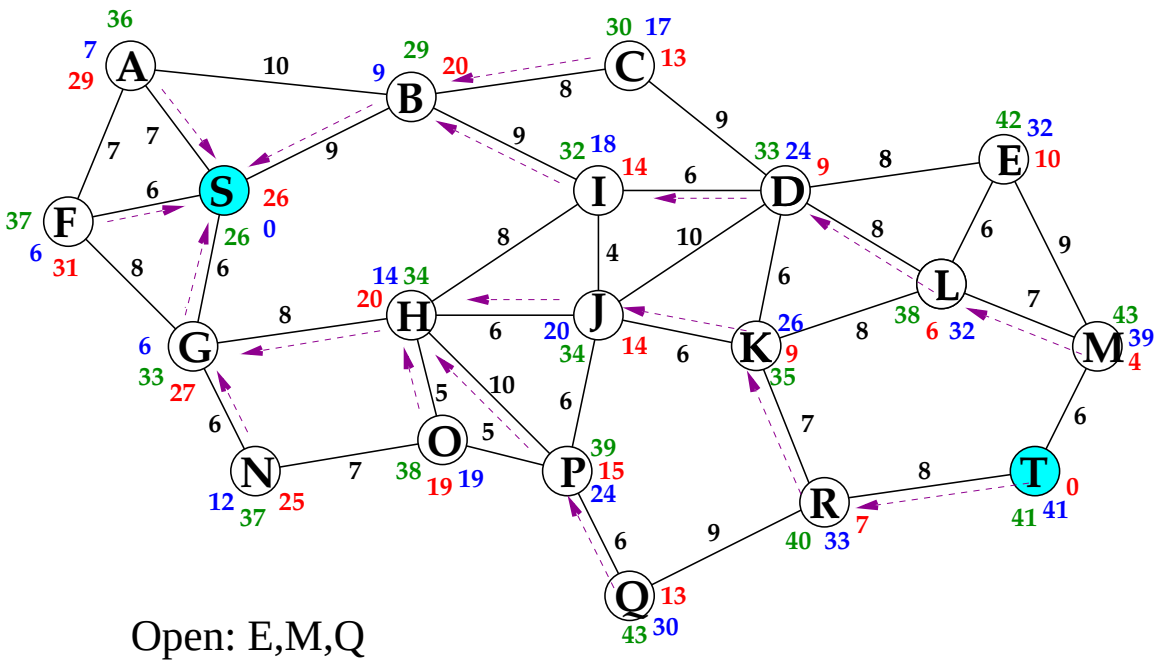
Open: L,O,P,R,E

Closed: S,B,C,I,D,G,H,J,K,A,F,N



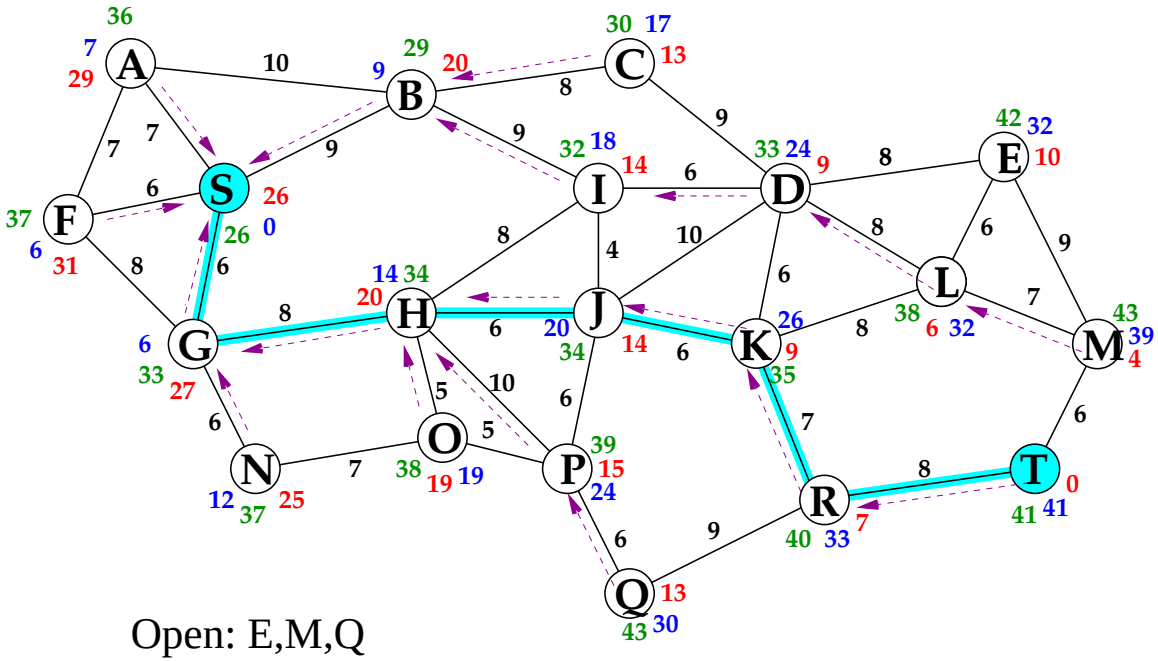
Open: P,R,E,M

Closed: S,B,C,I,D,G,H,J,K,A,F,N



Open: E,M,Q

Closed: S,B,C,I,D,G,H,J,K,A,F,N,P,R



Open: E,M,Q

Closed: S,B,C,I,D,G,H,J,K,A,F,N,P,R

21. Hashing

    (a) What is closed hashing?

    (b) What is open hashing?

    (c) What is open addressing?

    (d) What is a perfect hash function?

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

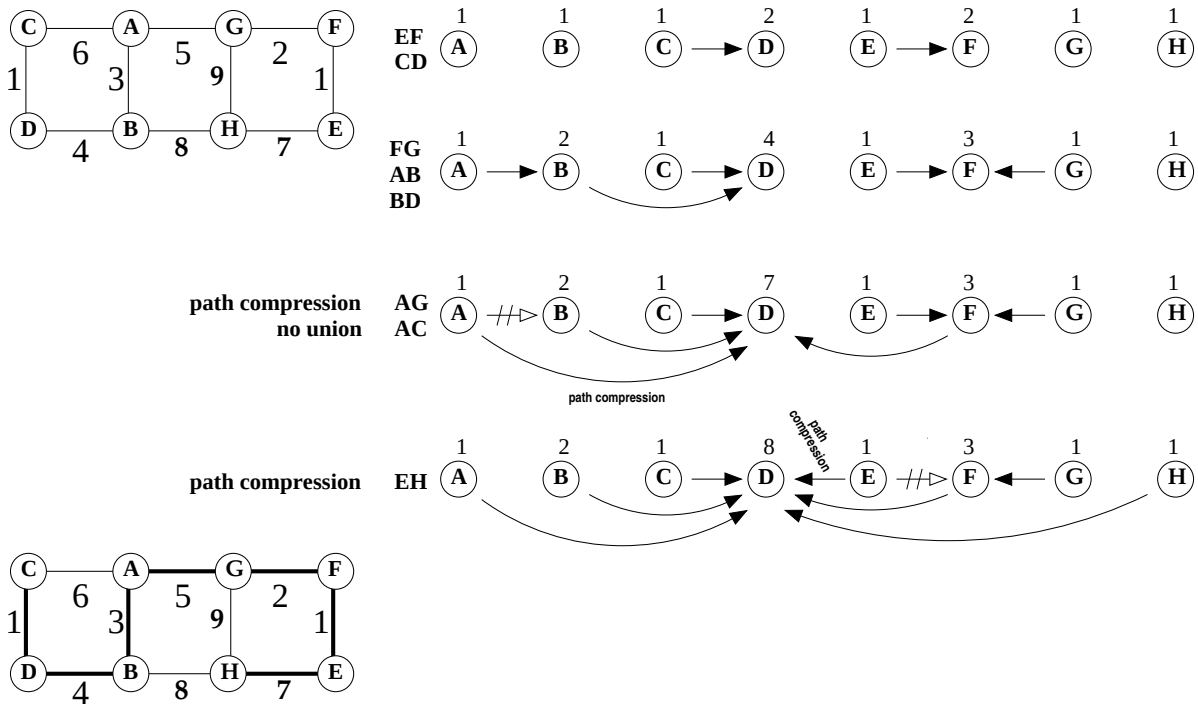
Must be deterministic.

Should be easy to calculate.

Should look random; in particular, should not have locality.

(f) What is cuckoo hashing?

22. 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.

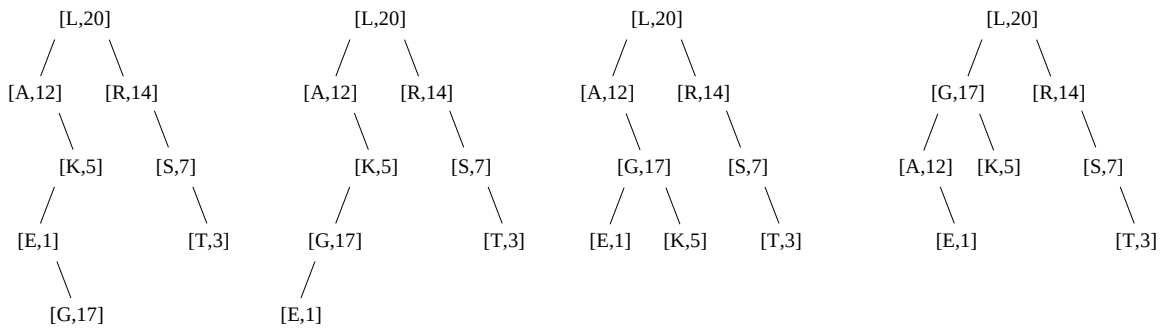


23. You are trying to construct a cuckoo hash table of size 8 holding 8 names. Each of the names listed below has two possible hash values, as indicated in the array. Put the items into the table in alphabetic order, if possible. Instead of erasing ejected items, simply strike them out.

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	~~BobGus~~ Jan
3	~~Ann~~ Bob
4	~~Dan~~ Hal
5	Fay
6	Dan
7	Ann

24. 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 <u>key</u>, a letter, and the second component is a the <u>priority</u>, a random integer. Insertion of the letter G, where the priority is chosen (at random) to be 17. Show the steps.



25. 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;
 }
```

Using either program, $f(n) = \Theta(n^2)$

The first subprogram uses recursion to compute $f(n)$ and takes $\Theta(n)$ time.

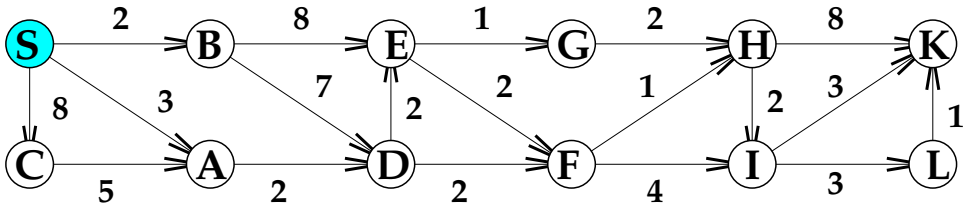
The second subprogram uses dynamic programming to compute $f(n)$ and takes $\Theta(n)$ time.

$f(n)$ can also be computed using memoization, taking $\Theta(\log^2 n)$ time.

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

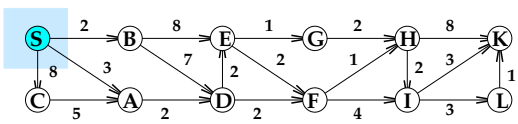
$-* \sim ab \wedge - \sim cde$

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

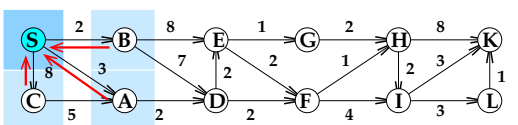


	S	A	B	C	D	E	F	G	H	I	K	L
V	0											
back												

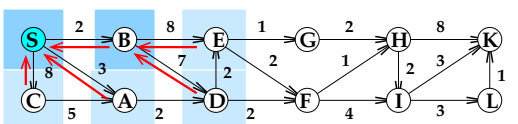
Fully processed nodes have a blue background.



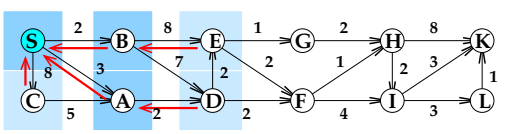
	S	A	B	C	D	E	F	G	H	I	K	L
V	0											
back												



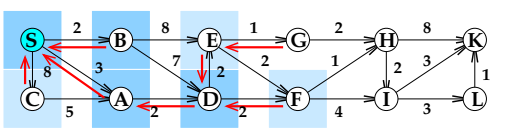
	S	A	B	C	D	E	F	G	H	I	K	L
V	0	3	2	8								
back		S	S	S								



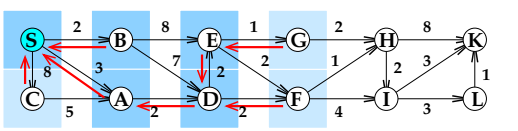
	S	A	B	C	D	E	F	G	H	I	K	L
V	0	3	2	8	9	10						
back		S	S	S	B	B						



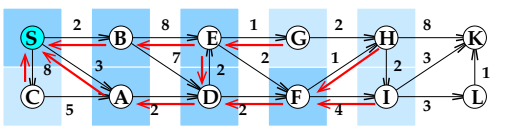
	S	A	B	C	D	E	F	G	H	I	K	L
V	0	3	2	8	~~9~~ 5	10						
back		S	S	S	~~B~~ A	B						



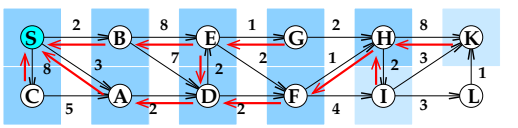
	S	A	B	C	D	E	F	G	H	I	K	L
V	0	3	2	8	~~9~~ 5	~~10~~ 7	7					
back		S	S	S	~~B~~ A	~~B~~ D	D					



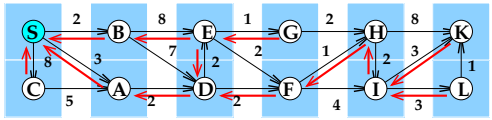
	S	A	B	C	D	E	F	G	H	I	K	L
V	0	3	2	8	~~9~~ 5	~~10~~ 7	7	8				
back		S	S	S	~~B~~ A	~~B~~ D	D	E				



	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		
back		S	S	S	~~B~~ A	~~B~~ D	D	E	F	F		



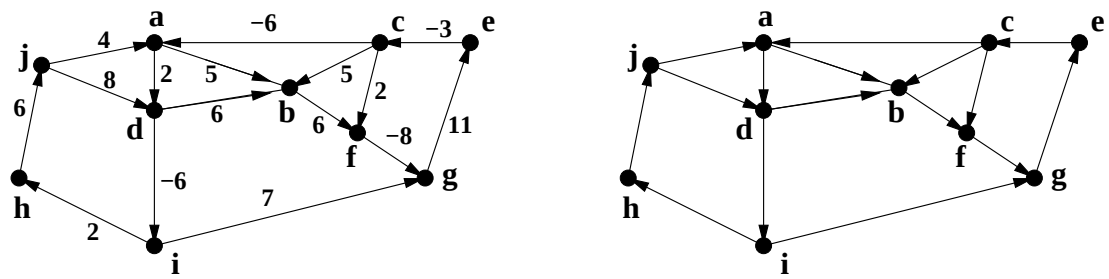
	S	A	B	C	D	E	F	G	H	I	K	L
V	0	3	2	8	~~9~~ 5	~~10~~ 7	7	8	8	~~11~~ 10	16	
back		S	S	S	~~B~~ A	~~B~~ D	D	E	F	~~F~~ H	H	



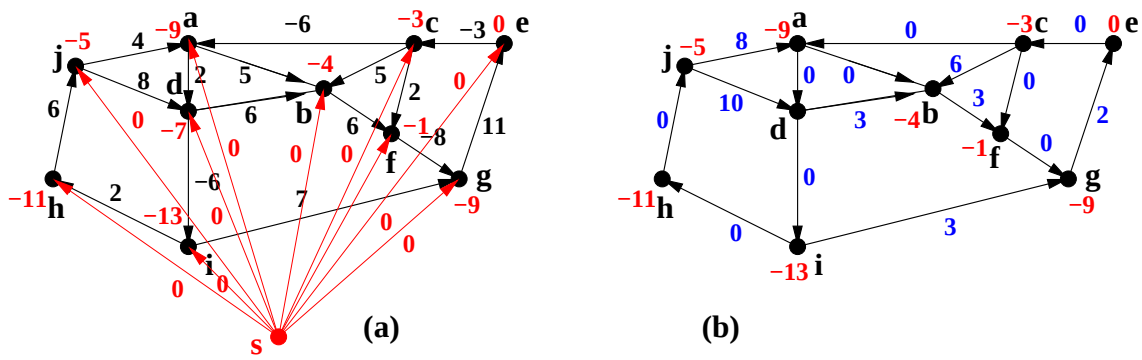
	S	A	B	C	D	E	F	G	H	I	K	L
V	0	3	2	8	~~9~~ 5	~~10~~ 7	7	8	8	~~11~~ 10	~~16~~ 13	13
back		S	S	S	~~B~~ A	~~B~~ D	D	E	F	~~F~~ H	~~H~~ I	I

28. 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.



(a) (b)



(a) (b)

29. Here is another coin-row problem. You have a row of coins of various values, where the value of the ith 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 ith 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.

    - 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.

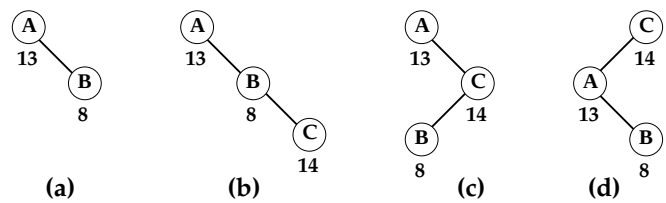
```
A[1] = V[1]
A[2] = V[2]
A[3] = V[3]
A[4] = V[4] + A[1]
A[5] = V[5] + max(A[1],A[2])
for i from 6 to n
```

```
A[i] = V[i] + max(A[i-5],A[i-4],A[i-3])
```

- 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.

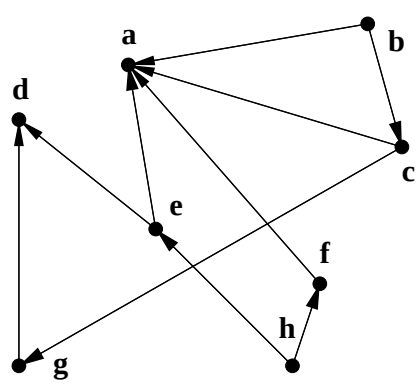
```
A[1] = V[1]
A[2] = max(V[2],A[1])
A[3] = max(V[3],A[2])
for i from 4 to n
  A[i] = max(A[i-1],V[i] + A[i-3])
```

30. 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.

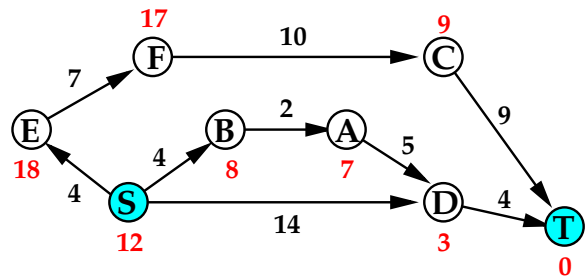


We first insert A, then B. Heap order is preserved, so no rotation is necessary. We next insert C. To restore heap order, we do a left roatation at B. Heap order is still not restored. We do a left rotation at A. Heap order is then restored.

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



32. 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
f	0							
g	12							
back								

Heap: BDE

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

Heap: DE

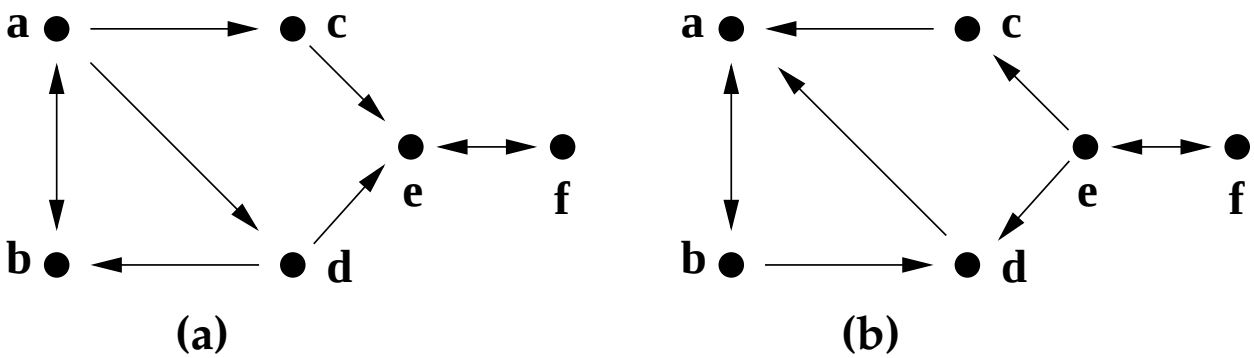
	S	A	B	C	D	E	F	T
h	12	7	8	9	3	18	17	0
f	0	6	4		11	4		
g	12	13	12		14	22		
back		B	S		A	S		

Heap: TE

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

33. Compute the strong components of the directed graph shown in (a), using the DFS method. Show the strong components in (b)