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

Assignment 4: Due Saturday October 24 2026 midnight

Follow our TA, Shareem Arjyal's (`arjyal@unlv.nevada.edu`) instructions on how to turn in the assignment.

Name:_____

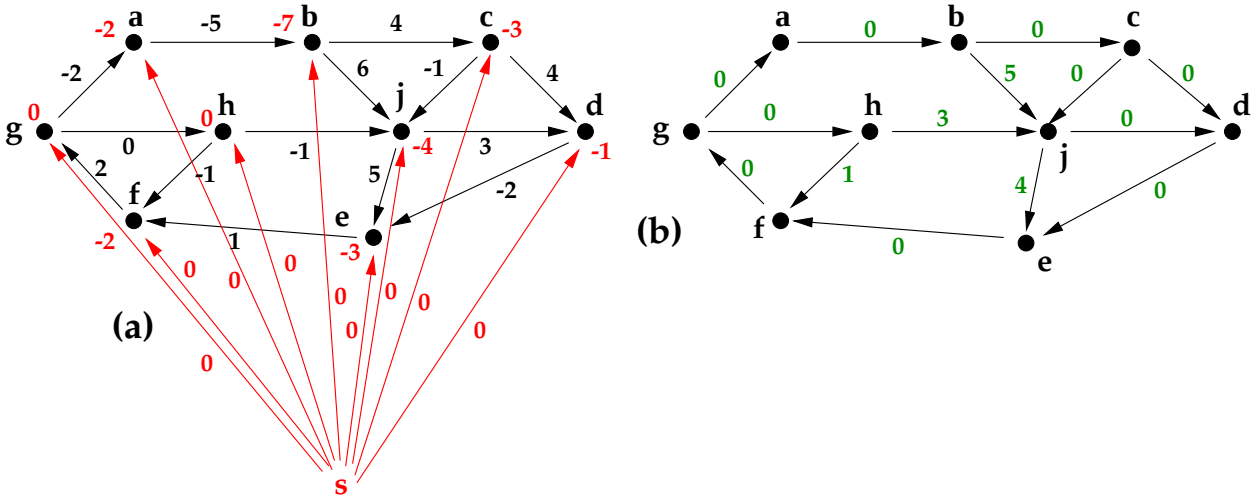
You are permitted to work in groups, get help from others, read books, and use the internet.

However, note that these friends and books, and the internet, will not be able to help you on the tests. I have no way of checking whether someone else did your homework, but if they did, you will not be able to pass the exams.

1. Fill in the blanks.

   (i) A **perfect** hash function has no collisions.

3. Work the first part of Johnson's algorithm on the weighted directed graph shown in (a) below. In (b) show the adjusted weight of each arc Do not finish Johnson's algorithm.



4. Here is C++ code which implements a recurrence which defines a function G.

```
int F(int n)
 {
  if(n <= 5) return 1;
  else return F(n/2) + F((n+1)/2 + n;
 }
```

Warning: The four questions below may have different answers.

(i) What is the asymptotic time complexity of a computation of $F(n)$, in terms of n, using the recursive C++ code above?

Asymptotically, $(n+1)/2 = n/2$: thus, we can use the recurrence
$T(n) = 2T(n/2) + 1$, since it takes $O(1)$ time to compute the last term, n. Thus $T(n) = O(n)$.

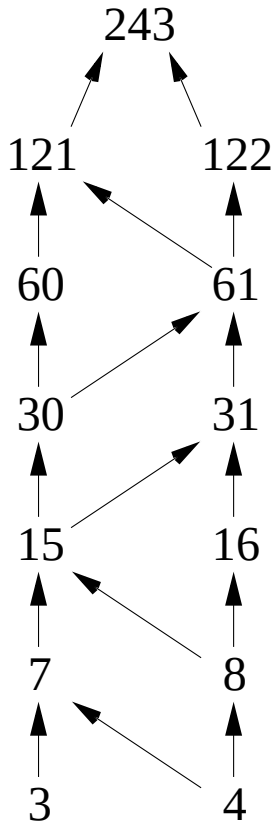
(ii) What is the asymptotic complexity of the function $F(n)$, in terms of n? (Not **time** complexity.)

Asymptotically, $(n+1)/2 = n/2$: thus, we can use the recurrence
$F(n) = 2F(n/2) + n$, hence $F(n) = \Theta(n \log n)$.

(iii) The function F could be computed using standard dynamic programming. What is the time complexity of such a computation of $F(n)$?

We need to compute $F(i)$ for all i from 1 to n, hence the time complexity is $O(n)$.

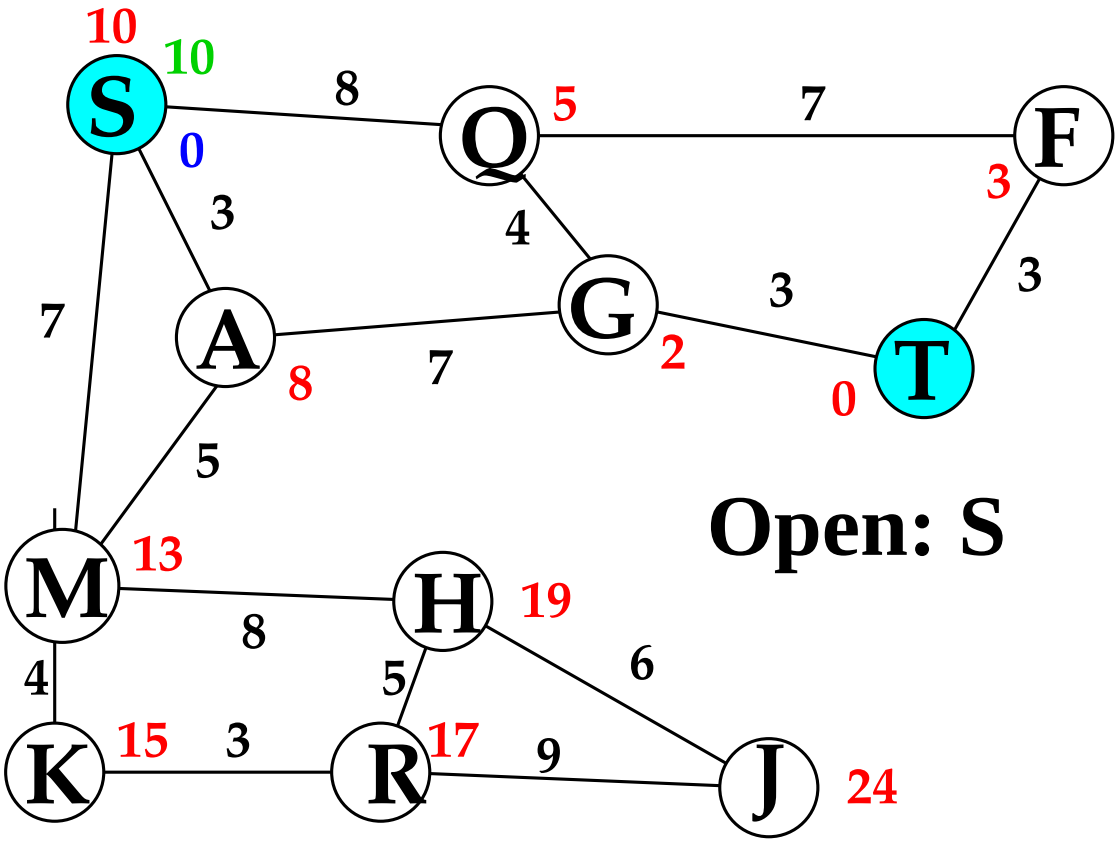
(iv) What is the asymptotic time complexity of a computation of $F(n)$ using memoization? Assume that each **fetch** or **store** takes $O(1)$ time.

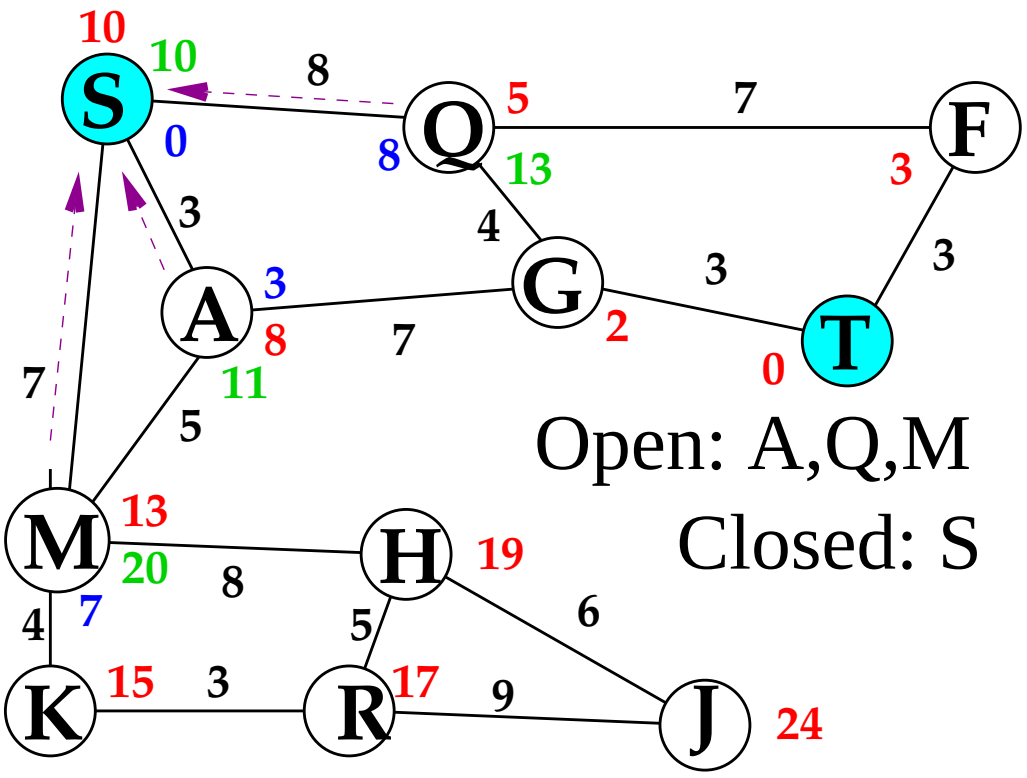


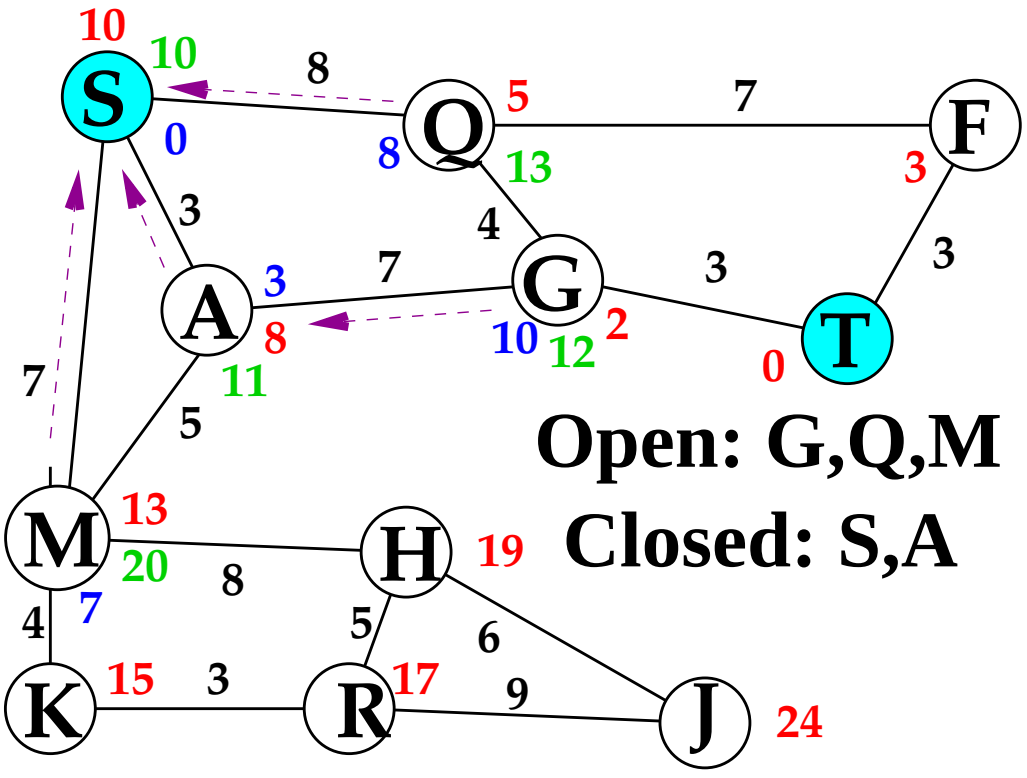
The subproblems do not grow exponentiall. In fact, we only need memos for $F(i)$ for $\Theta(\log n)$ distinct values of i, as shown here for $n = 243$. Hence the time complexity is $\Theta(\log n)$.

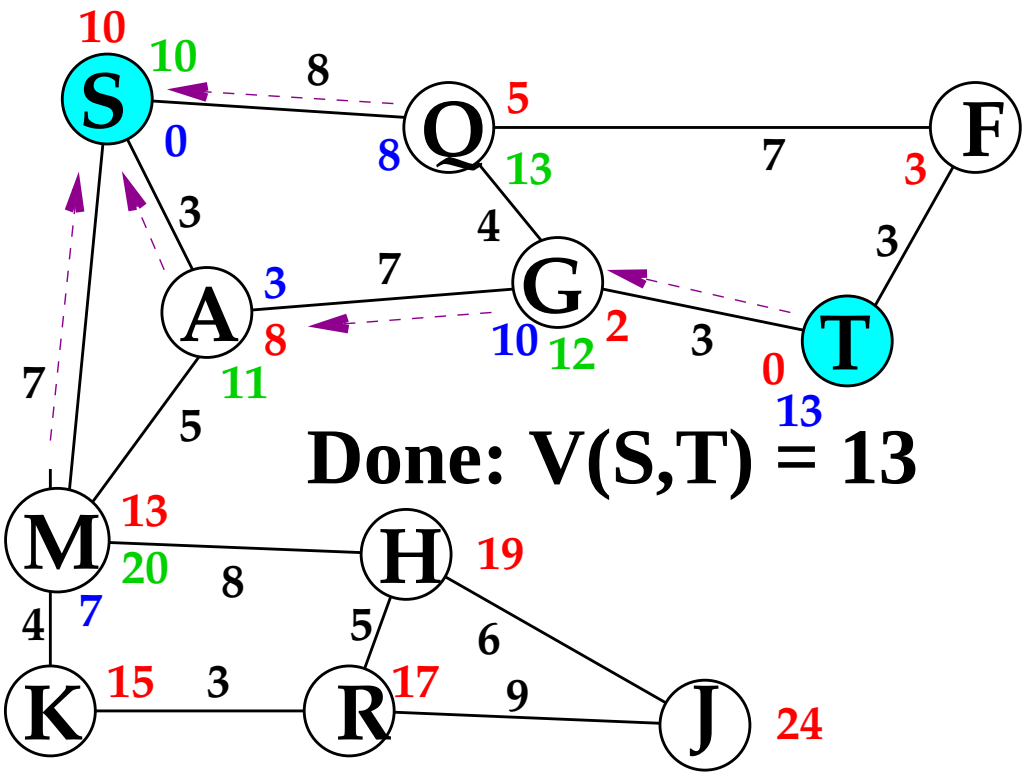
Walk throuth the A^* algorithm for the following weighted graph, finding the least cost path from S to T. The edge weights are in black and the heuristics are in red. The heuristics are both admissible and consistent. Your answer should label each fully processed vertex with both f and g values. Not all vertices will be processed.

Use copies of the graph on the next pages, as needed, to illustrate the steps.




10
10
S
8
Q
5
7
F
0
8
13
3
3
4
3
3
A
3
G
8
7
2
T
7
11
0
5
Open: A,Q,M
Closed: S
M
13
H
19
20
8
7
4
5
6
K
15
3
R
17
9
J
24


10
10
S
8
Q
5
7
F
0
8
13
3
3
4
3
3
A
3
7
G
8
10
12
2
T
7
11
0
5
Open: G,Q,M
Closed: S,A
M
13
H
19
20
8
7
4
5
6
K
15
3
R
17
9
J
24


10
10
S
8
Q
5
F
0
8
13
7
3
3
4
3
7
A
3
G
8
10
12
2
3
T
7
11
0
13
5
Done: V(S,T) = 13
M
13
H
19
20
8
7
4
5
6
K
15
3
R
17
9
J
24