

University of Nevada, Las Vegas Computer Science 477/677 Fall 2024

Answers to Assignment 3: Due Saturday October 19, 2026

1. Write C++ code for selection sort, using the elementary quadratic time on an array A of length n . Assume that A and n are given.

```
void swap(int&x, int&y)
{
    int z = x;
    int x = y;
    int y = z;
}

void sort()
{
    for(int i = 0; i < N; i++)
        for(int j = i; j < N-1; j++)
            if(A[j] < A[i]) swap(A[i],A[j]);
}
```

You must turn this part of the homework in in a form that the grader can compile and execute, definitely not **pdf**. *Follow the TA's instructions.*

2. Use **substitution** to find the asymptotic time complexity, in terms of n , for each of these code fragments. Show your steps, including explicitly stating your substitution. For example, you could say, "Let $m = \log n$."

```
for (int i = 1; i < n; i = 2*i)
    cout << "Hello, world."
```

Let $j = \log i$ and $m = \log n$. Then $\log 1 = 0$, and $\log 2i = \log 2 + \log i = 1 + j$, Substituting, we have:

```
for (int j = 0; j < m; j = j+1)
    cout << "Hello, world."
```

The time complexity is $\Theta(m) = \Theta(\log n)$.

3. This problem involves first one substitution, then another. Show the steps for each.

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

Let $j = \log i$ and $m = \log n$. Then $\log 2 = 1$, and $\log i * i = \log i + \log i = 2 \log i = 2 * j$, Substituting, we have:

```
for (int j = 1; j < m; i = 2*j)
    cout << "Hello, world."
```

Let $k = \log j$ and $p = \log m$. Then $\log 1 = 0$, and $\log 2j = \log 2 + \log j = 1 + k$, Substituting, we have:

```
for (int k = 0; k < p; k = k+1)
    cout << "Hello, world."
```

The time complexity is $\Theta(p) = \Theta(\log m) = \Theta(\log \log n)$.

4. The following two problems each use just one substitution, the same for each, but they have different answers. If you work the steps correctly, it will become clear why they have different answers.

(i)

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

Let $k = \log j$, thus $2*j = k + 1$. We obtain

```
for(int i = 1; i < n; i=i+1)
    for(int k = 0; k < log i; k=k+1)
        cout << "Hello, world."
```

The actual number of lines of output is the summation $\sum_{i=1}^{n-1} \sum_{k=0}^{\lceil \log i \rceil - 1} 1$ which is asymptotic to the

$$\text{integral} \int_1^n \int_0^{\ln x} dy dx = \int_1^n (\ln x) dx = \left[x \ln x - x \right]_1^n = \Theta(n \log n)$$

(Recall that $x \ln x - x$ is an anti-derivative of $\ln x$.)

(ii)

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

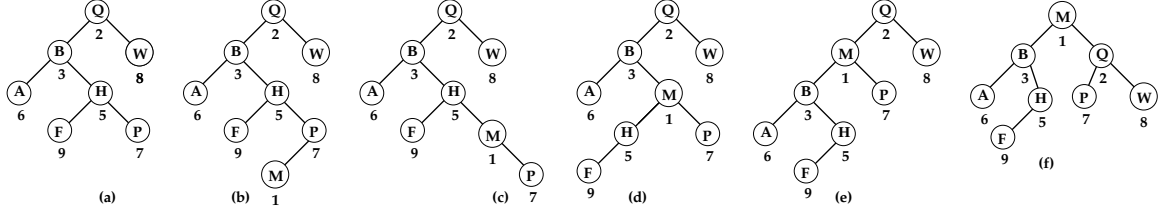
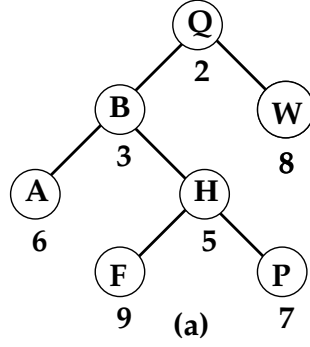
Use the same substitution as in the previous example. We obtain

```
for(int i = 1; i < n; i=i+1)
    for(int k = log i; k < log n; k=k+1)
        cout << "Hello, world."
```

$$\text{Approximated by the integral} \int_1^n \int_{\ln x}^{\ln n} dy dx = \int_1^n (\ln n - \ln x) dy dx = \left[x \ln n - x \ln x + x \right]_1^n = \Theta(n)$$

5. The figure below shows a treap containing six items, whose priorities are given in the table below. In a sequence of figures, show the insertion of a new item, M, with priority 1. Your sequence should show all the rotations. Your treap must use min-heap order.

item	priority
A	6
B	3
F	9
H	5
Q	2
W	8



6. Solve each of these recurrences, giving each answer as a function of n using Θ notation. In each case, if you have to guess the value of the exponent δ , it will be either an integer or a simple fraction.

Recall that the last term is n^γ . The solution is $\Theta(n^\gamma \log n)$ if $\sum \alpha_i (\beta_i)^\gamma = 1$, $\Theta(n^\gamma)$ if $\sum \alpha_i (\beta_i)^\gamma < 1$, otherwise it is n^δ for $\sum \alpha_i (\beta_i)^\delta = 1$.

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

$\gamma = 1$ and $(1/3)^1 + (2/3)^1 = 1$, hence $F(n) = \Theta(n \log n)$.

(ii) $F(n) = F(4n/5) + F(3n/5) + n^2$

$\gamma = 2$ and $(4n/5)^2 + (3n/5)^2 = 1$, hence $F(n) = \Theta(n^2 \log n)$.

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

$\gamma = 0$ and $3 \cdot (1/3)^0 + 3 \cdot (2/3)^0 = 6 > 1$, hence we need find δ such that $3(1/3)^\delta + 3(2/3)^\delta = 1$. The correct value is $\delta = 3$. Thus, $F(n) = \Theta(n^\delta) = \Theta(n^3)$.

7. No one got this correct on the first exam, but I was **extremely** generous in my grading, so some students got full credit. However, the TA is instructed to take off for even very small errors while grading this problem.

Here is a linked list implementation of stack of integer using C++.

```
struct listnode;
typedef listnode*stak;
struct listnode
{
    int itm;
    stak pntr;
};

bool empty(stak s)
{
    // returns true if s is the empty stack, otherwise false
    return s == nullptr;
}

void push(sta & s, int n)
{
    stak newstak = new listnode;
    newstack->itm = n;
    newstack->ntr = s;
    s = newstak;
}

stak pop(stak & s)
//
{
    assert(not empty(s)); // alternatively, you could write assert(s);
    int tmp = s->itm;
    s = s->pntr;
    return tmp;
}
```

8. Write pseudocode for the Floyd Warshall algorithm. I will instruct the TA to be strict while grading this problem.

Assume that an array A of length n is given to us, as well as the value $W[i, j]$ of the arc from i to j , for all vertices i and j , and that that value is ∞ if there is no such arc. We construct arrays V and back , where $V[i, j]$ is the least weight of any path from i to j , and $\text{back}[i, j]$ is the backpointer of that path

```

For all  $i$  from 1 to  $n$ 
  For all  $j$  from 1 to  $n$ 
     $V[i, j] = W[i, j]$ 
     $\text{back}[i, j] = i$ 
For all  $i$  from 1 to  $n$ 
   $V[i, i] = 0$ 
For all  $j$  from 1 to  $n$ 
  For all  $i$  from 1 to  $n$ 
    For all  $k$  from 1 to  $n$ 
       $\text{temp} = V[i, j] + V[j, k]$ 
      if( $\text{temp} < V[i, k]$ )
         $V[i, k] = \text{temp};$ 
         $\text{back}[i, k] = \text{back}[j, k]$ 

```

9. The basic “coin-row” problem is to select a maximum value “legal” set from a given row of coins, where each coin has a given value and it is illegal to select any two coins which are adjacent in the row.

Write a C++ program which solves this problem. You may assume that there are N coins in the row, and $X[i]$ is the value of the i^{th} coin. Use dynamic programming. Assume that the value of each coin is a positive integer.

There are two ways to solve this program using dynamic programming. The first way is to let $A[i]$ be the maximum total of any legal subset of the first i coins:

```

int A[n];

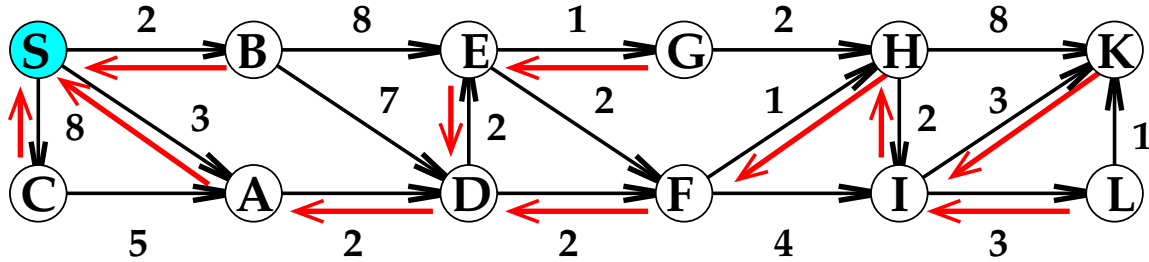
A[0] = 0;
A[1] = X[1];
for(int i = 2; i <= N; i++)
  A[i] = max(A[i-1], X[i]+A[i-2]);
cout << "The maximum value of any legal set is" << A[N] << endl;

```

Of course, that code does not give you the actual best set. To do that, you need to introduce backpointers. You must turn this part of the homework in in a form that the grader can compile and execute, definitely not **pdf**. *Follow the TA's instructions.*

10. Walk through the steps of Dijkstra's algorithm for the following weighted directed graph. The source node is S .

You do not need to redraw the graph or make a new table at each step. Write entries at each step. If an entry is updated, do not erase it; instead, simply cross it out.



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