University of Nevada, Las Vegas Computer Science 477/677 Fall 2022

Answers to Assignment 4: Due Friday October 14, 2022

1. Fill in the blanks. One word per blank.

   (a) The two standard operators of the ADT *stack* are **push** and **pop**.

   (b) The two standard operators of the ADT *array* are **fetch** and **store**.

2. Find the asymptotic time complexity of each code fragment. Give the answer using Θ notation.

   (a)
   ```
   for(int i = 0; i*i < n; i++)
   ```
   $\Theta(\sqrt{n})$

   (b)
   ```
   for(float x = n; x > 2.0; x = sqrt(x))
   ```
   $\Theta(\log\log n)$

   (c)
   ```
   for(int i = 1; i < n; i = 2*i)
    for(int j = 2; j < i; j = j*j);
   ```
   $\Theta(\log n \log\log n)$

3. Solve the recurrences. Give asymptotic answers in terms of n.

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

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

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

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

   (c) $F(n) = 2F(n/2) + n$

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

   (d) $F(n) = n + F(n/5) + F(7n/10)$

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

   (e) $F(n) = F(3n/4) + F(n/2) + 3F(n/4) + n$

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

   (f) $F(n) = F(n-1) + \sqrt{n}$

   $F(n) = \Theta\left(n^{\frac{3}{2}}\right)$

(g) $F(n) = F(n - \log n) + \log n$

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

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

$F(n) = \log^* n$

(i) $F(n) = 2F(n/2) + 2F(n/6) + F(2n/3) + n^2$

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

4. Consider the function f(n) computed by the code below.

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

The function can be computed by recursion, as given in the C++ code above. However, we could also compute f(n) using dynamic programming or memoization.

(a) What is the asymptotic value of f(n)? The value itself, not the time to compute it. Write the recurrence, then solve it.

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

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

(b) What is the asymptotic time complexity of the recursive computation of f(n), using the code given? Write the recurrence, then solve it.

$T(n) = T(n/2) + TF(n/4) + 1$

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

(c) What is the asymptotic time complexity of the dynamic programming computation of f(n)?

$\Theta(n)$

(d) What is the asymptotic time complexity of the computation of f(n) using memoization?

$\Theta(\log n)$

5. You are given an array A of N integers, where N could be any positive integer. Write C++ code that prints out the sum of the entries of A, then write C++ code that prints out the maximum entry of A. The code must be correct C++ code.

```
int const N = 1000; // or whatever
int A[N];

int main()
 {
  int sumA = 0;
  for(int i = 0; i < N; i++)
   sumA = sumA + A[i];
  int maxA = A[0]; // you can’t initialize to any constant
  for(int i = 1; i < N; i++)
   if(A[i] > maxA)
    maxA = A[i];
  cout << "The sum of the array A is " << sumA;
  cout << "The maximum of the array A is " << maxA;
  return 1;
 }
```

6. You are given a sequence X of N positive numbers. Give an algorithm which finds the maximum total subsequence, subject to the condition that the subsequence contains no two consecutive terms of the original sequence. Write your algorithm in correct C++ code.

```
int const N = 20; // or whatever
int X[N]; // input sequence (global)
int back[N]; // back pointers (global)
int maxlegal[N]; // maximum total legal subsequence ending at i

void printsubsequence(int last)
 {
  if(last <= 1) cout << X[last];
  else
   {
    printsubsequence(back[last]);
    cout << "," << X[last];
   }
 }

int main()
 {
  for(int i = 0; i < N; i++)
   if(i == 0) maxlegal[0] = X[0];
   else if(i == 1) maxlegal[1] = X[1];
   else if(i == 2)
    {
     maxlegal[2] = X[0]+X[2];
     back[2] = 0;
    }
   else
    {
     int j;
     if(X[i-3] > X[i-2]) j = i-3;
     else j = i-2;
     maxlegal[i] = maxlegal[j] + X[i];
     back[i] = j;
    }
  cout  << " " << endl;
  if(N == 1 or maxlegal[N-1] > maxlegal[N-2])
   printsubsequence(N-1);
  else
   printsubsequence(N-2);
  cout << endl;
  return 1;
}
```