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

Answers to Assignment 4: Due Friday October 11, 2024

1. Name three kinds of priority queue.

   stack, queue, heap.

2. Solve the following recurrences.

   (a) $F(n) = F(n-1) + \log n$

   $F'(n) \approx \log n$

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

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

   $\dfrac{F(n) - F(n - \log n)}{\log n} = 1$

   $F'(n) \approx 1$

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

   (c) $F(n) = 2F(n/5) + \sqrt{n}$

   $A = 2, B = 5, C = \frac{1}{2}\ B^C = \sqrt{5} > A$

   $F(n) = \Theta(\sqrt{n})$

   (d) $F(n) = 3F(n/9) + 1$

   $A = 3, B = 9, C = 0\ B^C = 1 < A$

   $F(n) = \Theta(n^{\log_9 3}) = \Theta(n^{\frac{1}{2}}) = \Theta(\sqrt{n})$

   (e) $F(n) = 2F(n-1) + 1$

   Let $n = \log m$ and $G(m) = F(n)$. Note that $n - 1 = log(m/2)$, thus $G(m/2) = F(n-1)$ We have:

   $G(m) = 2G(m/2) + 1$

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

   (f) Up to now, no student has correctly analyzed this next recurrence.

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

   We'll wait until some student analyzes it correctly.

3. Write pseudocode for the Floyd-Warshall algorithm. Let the vertices be named by integers 1 through n. Let $W[i, j]$ be the weight of the arc from i to j, ∞ if there is no such arc. The output should consist of two arrays,

   (a) $V[i, j]$, the weight of the least weight path from i to j.

   (b) BACK$[i, j]$, the back pointer of that path.

```
for i from 1 to n
  for j from 1 to n
     V[i,j] = W[i,j]
     back[i,j] = W[i,j]
for i from 1 to n
  V[i,i] = 0
for j from 1 to n
  for i from 1 to n
    for k from 1 to n
      int temp = V[i,j] + V[j,k]
      if(temp < V[i,k])
        V[i,k] = temp
      back[i,k] = back[j,k]
```

4. Write pseudocode for the Bellman-Ford algorithm. Let the vertices be named by integers 0 through n, where 0 is the source vertex. Let $W[i,j]$ be the weight of the arc from i to j, ∞ if there is no such arc. The output should consist of two arrays,

   (a) $V[i]$, the weight of the least weight path from 0 to i.

   (b) BACK$[i]$, the back pointer of that path.

   Your code should include the shortcut to end computation when the solution is complete.

```
V[0] = 0;
for i = 1 to n
  V[i] = W[0,i]
  back[i] = 0;
bool changed = true
while(changed)
  changed = false
  for i = 1 to n
    for j = 1 to n
      temp = V[i] + W[i,j]
      if(temp < V[j])
        V[j] = temp
        back[j] = i
        changed = true
```

5. What is the asymptotic time complexity, in terms of n, of this recursive C++function? Can you analyze it using a recurrence? (Yes, you can.)

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

6. The following problem is called a "coin-row" problem. Given a row of coins of various values, select the set of coins of maximum value, subject to the rule that you may not select any two adjacent coins. For example, if the values are 1,5,8,6,1,2,7,3 the maximum is achieved by selecting the coins worth 5, 6, 7.

   Write a C++ program which solves this problem for an arbitrary sequence of values. You may assume that the values are already given. The only part of the program you need to write is the function which computes the best set of coins to select. Miss Farivar will give you more precise instructions if needed.

```
const int n = 20; // number of coins
const int sentinel = -1;
int coin[n];
int A[n]; // A[i] = maximum sum of legal subsequence ending at coin[i];
int B[n];  // B[i] = backpointer

void getcoins()
 {
  for(int i = 0; i < n; i++)
   cin >> coin[i];
 }

void computeA()
 {
  A[0] = coin[0];
  B[0] = sentinel;
  A[1] = coin[1];
  B[1] = sentinel;
  A[2] = coin[0]+coin[2];
  B[2] = 0;
  for(int i = 3; i < n; i++)
   {
    int a1 = coin[i] + A[i-3];
    int a2 = coin[i] + A[i-2];
    //cout << "i = " << i << " a1 = " << a1 << " a2 = " << a2 << endl;
    if(a1 > a2)
     {
      A[i] = a1;
      B[i] = i-3;
     }
    else
     {
      A[i] = a2;
      B[i] = i-2;
     }
   }
 }
```

```
void writecoins()
 {
  for(int i = 0; i < n-1; i++)
   cout << coin[i] << ",";
  cout << coin[n-1] << "." << endl;
 }

void writebest(int n)
 {
  if(n > sentinel)
   {
    writebest(B[n]);
    cout << coin[n] << " + ";
   }
 }

void writebest()
 {
  if (A[n-1] > A[n])
   {
    writebest(B[n-1]);
    cout << coin[n-1] << " = " << A[n-1] << endl; }
  else
   {
    writebest(B[n]);
    cout << coin[n] << " = " << A[n-1] << endl;
   }
 }

int main()
 {
  getcoins();
  writecoins();
  computeA();
  writebest();
  return 1;
 }
```

My input: 56 4 9 21 4 78 65 7 44 83 25 1 90 86 92 93 41 50 24 37.

My output:

```
56,4,9,21,4,78,65,7,44,83,25,1,90,86,92,93,41,50,24,37.
56 + 21 + 78 + 7 + 83 + 90 + 93 + 50 + 37 = 515
```