1. What is the worst case asymptotic time complexity for each of these shortest path algorithms?

(a) Bellman Ford. $O(nm)$
(b) Floyd Warshall. $\Theta(n^3)$
(c) Dijkstra. $O(m \log n)$
(d) Johnson. $O(nm \log n)$

2. Write pseudocode for the Floyd Warshall algorithm. Let $W[i,j]$ be the length of the arc from $i$ to $j$, which could be $\infty$.

The program computes $V[i,j]$, the length of the shortest path from $i$ to $j$ and the backpointer $B[i,j]$. $B[i,i]$ is undefined, since the shortest path from $i$ itself has no arcs.

\begin{verbatim}
for(i = 1 to n)
  for(j = 1 to n)
    V[i,j] = W[i,j]
    B[i,j] = i
for i = 1 to n
  V[i,i] = 0
for j = 1 to n
  for i = 1 to n
    for k = 1 to n
      temp = V[i,j] + V[j,k]
      if(temp < V[i,k])
        V[i,k] = temp
        B[i,k] = B[j,k]
\end{verbatim}

\item Write pseudocode for the Bellman Ford algorithm. Be sure to include the shortcut.

\begin{verbatim}
V[0] = 0
for i = 1 to n
  V[i] = \infty
  B[i] = -1
for i = 1 to n
  for k = 1 to m
    if(W[i,k] < V[i])
      V[i] = W[i,k]
      B[i] = k
\end{verbatim}
V[i] = infinity
finished = false
t = 0
while(not finished and t < n)
{
    finished = true;
t++
    for(k = 1 to m)
        i = s[k]
j = t[k]
temp = V[i] + W[k]
    if(temp < V[j])
        V[j] = temp
        B[j] = i
        finished = false
}

The purpose of the variable t is to prevent the program from running forever if the graph contains a negative weight cycle.
3. Walk through Dijkstra’s algorithm for the single source minpath problem for the directed graph illustrated on the next page. Instead of numbering the vertices 0 through 19, I have assigned them letters from A to T. The source vertex is S.

After each iteration of the main loop, show
1. The array dist, where dist[x] is the smallest length of any path found so far from S to x. (Initially, dist[x] = ∞ for most x.)
2. The array back, where back[x] is the next-to-the last vertex on the path of smallest weight found so far from S to x.
3. The contents of heap. Do not try to show the structure of the heap, simply list its members.

|   | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | heap |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| dist back |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
Continue your work on Problem 3 on this page.

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | heap |
| dist | 7 | 10 | 6 | 6 | 14 | 12 | | | | | | | | | | | | | | | | | BNH |
| back | S | S | S | | | | | | | | | | | | | | | | | | | | | NHCI |
| dist | 7 | 10 | 6 | 6 | 14 | 12 | | | | | | | | | | | | | | | | | | CIOJP |
| back | S | S | S | S | | | | | | | | | | | | | | | | | | | | IOJP |
| dist | 7 | 10 | 6 | 6 | 14 | 12 | 19 | | | | | | | | | | | | | | | | JPD |
| back | S | S | S | S | S | | | | | | | | | | | | | | | | | | PDK |
| dist | 7 | 10 | 6 | 6 | 14 | 12 | 19 | 14 | 19 | 24 | | | | | | | | | | | | KQEL |
| back | S | S | S | S | S | | | | | | | | | | | | | | | | | | REL |
| dist | 7 | 10 | 6 | 6 | 14 | 12 | 19 | 14 | 19 | 24 | 0 | 105 | | | | | | | | | | | TM |
| back | S | S | S | S | S | | | | | | | | | | | | | | | | | | M |

Backpointers are indicated.
4. Write C++ code (which must be able to be executed) for a dynamic program which finds the maximum weight legal subsequence of a sequence of positive integers. A subsequence is legal if it has no consecutive terms of the input sequence. Your program should use backpointers to recover the best subsequence.

In order to make the grader's task easier, here are two sequences I want you to use while testing your program, each of length 20.

15,324,184,48,102,31,119,15,26,160,129,78,14,7,64,83,22,88,185,123.

Remember that the task of this assignment is not just to find the correct answers, but to write and run a correct program.

I have given you two algorithms for this problem. Here is one of them: //

```cpp
#include <cstdio>
#include <iomanip>
#include <cassert>
#include <string>
#include <cmath>
#include <iostream>
#include <sstream>
#include <stdio.h>
#include <stdlib.h>
using namespace std;

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

//

Here is the other //

#include<stdio>
#include<iomanip>
#include<cassert>
#include<string>
#include<cmath>
#include<iostream>
#include<sstream>
#include<stdlib.h>
using namespace std;

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] = next to last coin of max legal subseq ending at coin[i];

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;
}
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
    {

However, the programs are equivalent, that is, with the same input they give the same output. Here are runs with the two input sequences I gave you.

39 + 53 + 23 + 298 + 59 + 116 + 130 + 195 + 37 + 206 = 1196

15, 324, 184, 48, 102, 31, 119, 15, 26, 160, 129, 78, 14, 7, 64, 83, 22, 88, 185, 123.
324 + 102 + 119 + 160 + 78 + 7 + 64 + 83 + 88 + 123 = 1084