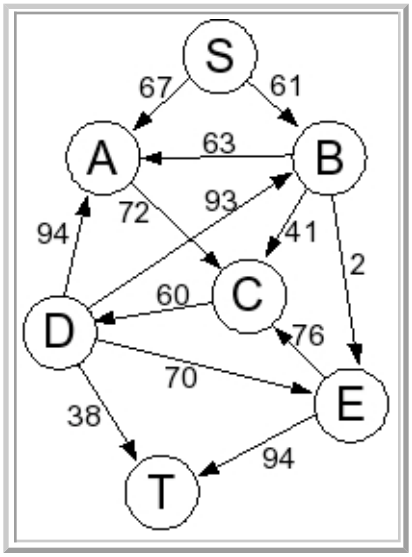
CS302 -- Final Exam. May 1, 2008

Please, write your answers on a separate sheet, not on the exam.

Remember your name, too...

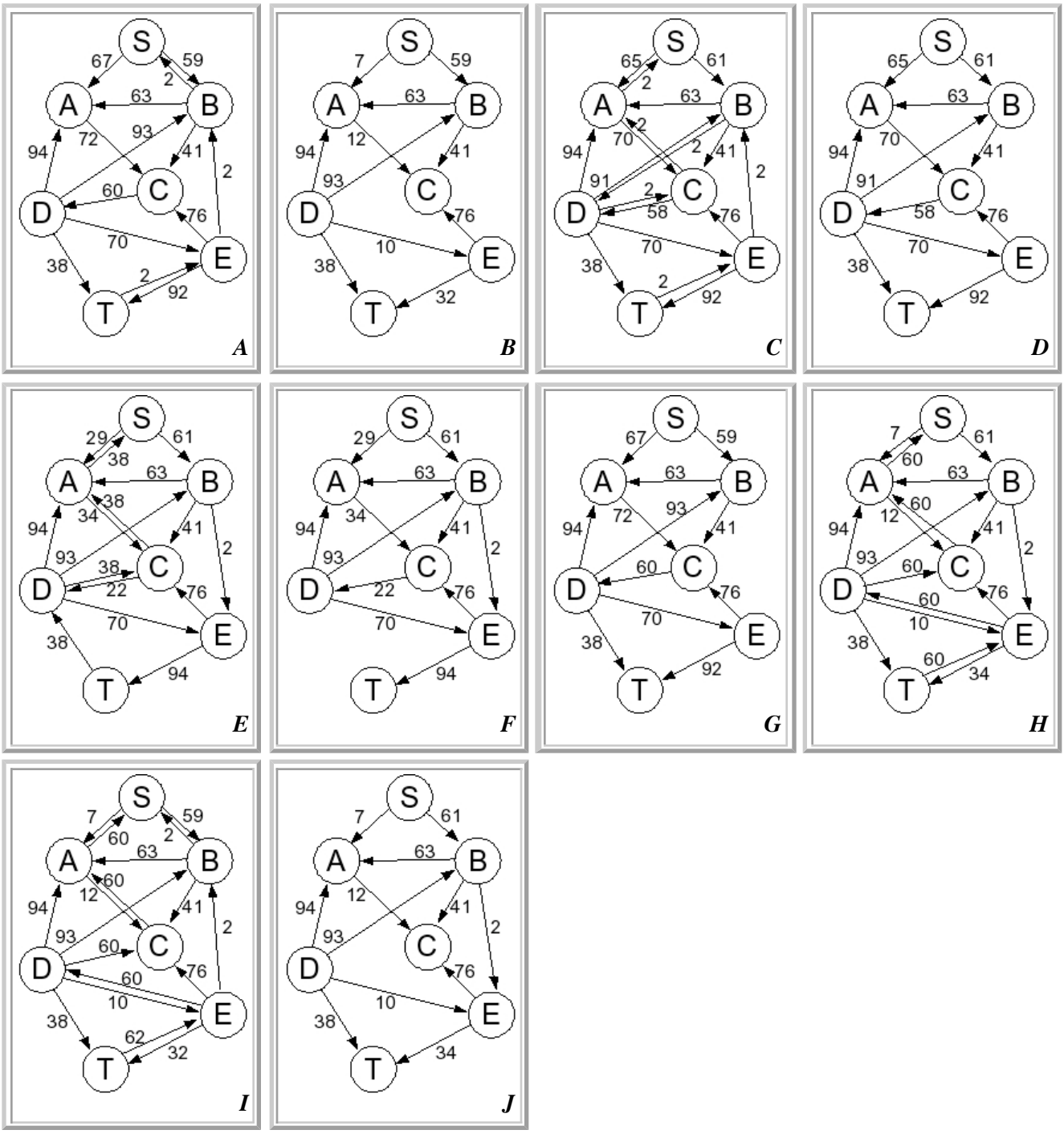
Question 1 (45 minutes)

Behold the following graph:

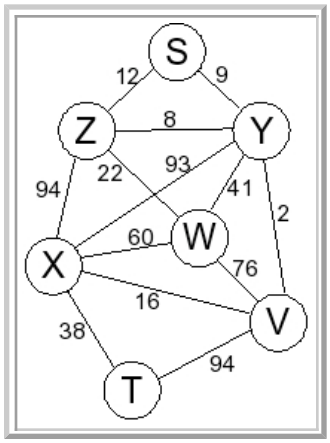


On the next page, there are ten graphs. You are to answer the following, using those ten graphs when necessary.

- **Part A:** Which of the ten graphs is a final residual flow graph that would result when an augmenting path network flow algorithm runs on our graph.
- **Part B:** What is the value of the maximum flow of through the graph?
- **Part C:** What is the minimum cut of the graph?
- **Part D:** Which of the ten graphs is the residual flow graph after one step of the Edmonds-Karp algorithm?
- **Part E:** Which of the ten graphs is the residual flow graph after one step of the greedy depth-first search algorithm?
- **Part F:** Which of the ten graphs is the residual flow graph after one step of the algorithm that uses the modified Dijkstra's algorithm to find the augmenting path?
- **Part G:** Describe how you use the answer to part **A** to come up with the answer to part **C**. Be specific.


S
A
B
C
D
E
T
67
59
2
63
72
93
94
41
2
60
76
70
38
2
92
A
S
A
B
C
D
E
T
7
59
63
12
94
93
41
10
38
32
B
S
A
B
C
D
E
T
65
61
2
63
70
2
94
91
2
41
2
58
76
70
38
2
92
C
S
A
B
C
D
E
T
65
61
63
70
94
91
41
58
76
70
38
92
D
S
A
B
C
D
E
T
29
61
38
63
38
34
94
41
2
93
38
22
76
70
38
94
E
S
A
B
C
D
E
T
29
61
63
34
94
41
2
93
22
76
70
94
F
S
A
B
C
D
E
T
67
59
63
72
93
94
41
60
76
70
38
92
G
S
A
B
C
D
E
T
7
61
60
63
60
12
94
41
2
93
60
76
60
10
38
60
34
H
S
A
B
C
D
E
T
7
59
2
60
63
60
12
94
41
2
93
60
76
60
10
38
62
32
I
S
A
B
C
D
E
T
7
61
63
12
94
93
41
10
38
34
J

Question 2 (15 minutes)



Using the graph drawn to the right, answer the following:

- **Part A:** Draw the minimum spanning tree of this graph.
- **Part B:** If one used Prim's algorithm starting at node **S**, give the order in which edges are added when constructing the minimum spanning tree. If an edge is not part of the minimum spanning tree, do not include it.
- **Part C:** If one used Kruskal's algorithm, give the order in which edges are added when constructing the minimum spanning tree. If an edge is not part of the minimum spanning tree, do not include it.

Question 3 (30 minutes)

Here's a partially-completed implementation of Dijkstra's shortest path algorithm from one node to another:

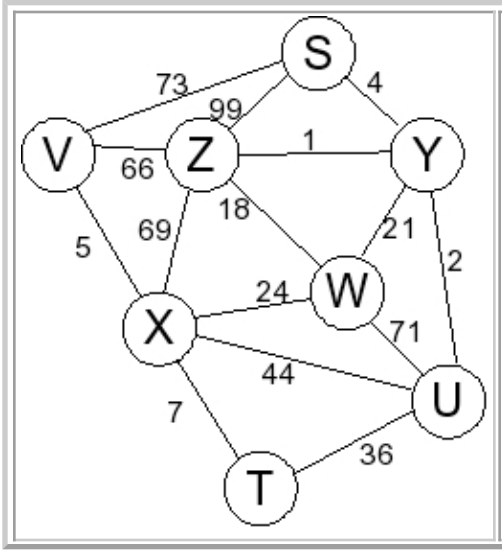
```
#include <map>
#include <queue>
#include <deque>
#include <iostream>
#include <string>
using namespace std;

class Node {
  public:
    string name;
    int distance;
    Edge *backedge;
    queue <class Edge *> edges;
};

class Edge {
  public:
    int weight;
    Node *from;
    Node *to;
};

class Graph {
  public:
    deque <Edge *> *
       Shortest_Path(Node *from, Node *to);
    vector <Node *> nodes;
    multimap <int, Node *> Dijkstra;
};
```

```
deque <Edge *> *
   Graph::Shortest_Path(Node *from, Node *to)
{
  deque <Edge *> *path;
  int i;
  Node *n;
  multimap <int, Node *>::iterator dit;

  for (i = 0; i < nodes.size(); i++) {
    nodes[i]->distance -1;
    nodes[i]->backedge = NULL;
  }
  from->distance = 0;
  Dijkstra.insert(make_pair(0, from));

  while (!Dijkstra.empty()) {
    dit = Dijkstra.begin();
    n = dit->second;
    Dijkstra.erase(dit);
    if (n == to) {
      path = new deque <Edge *>;
      while (n != from) {
        path->push_front(n->backedge);
        n = n->backedge->from;
      }
      return path;
    }
    // The rest of Dijkstra's algorithm goes here
  }
  return NULL;
}
```

Now, consider the graph below, left. Suppose we construct an instance of **Graph** holding this graph, and

we call **Shortest_Path** from **S** to **V**.

When the **while()** loop begins, the state of **Graph** is pictured below, right.



Nodes:

S	0		[(4,S,Y),(99,S,Z),(73,S,V)]
V			[(73,V,S),(66,V,Z),(5,V,X)]
Z			[(99,Z,S),(66,Z,V),(1,Z,Y),(18,Z,W),(69,Z,X)]
Y			[(4,Y,S),(1,Y,Z),(21,Y,W),(2,Y,U)]
X			[(5,X,V),(69,X,Z),(24,X,W)(44,X,U),(7,X,T)]
W			[(24,W,X),(18,W,Z),(21,W,Y),(71,W,U)]
T			[(7,T,X),(36,T,U)]
U			[(2,U,Y),(71,U,W),(44,U,X),(36,U,T)]

Dijkstra: (0,S)

- **Part A**: Draw the state of **Graph** after the first iteration of the **while()** loop. The answer sheet has a form for you to fill in -- fill in all blank parts.
- **Part B**: Draw the state of **Graph** after the second iteration of the **while()** loop.

Question 4 (30 minutes)

You are given a two-dimensional vector of doubles, **A**. Suppose *i* < **A.size()** and suppose *j* < **A[0].size()**. Then we define the function *f(i,j)* as follows:

- *f(0,0)* = **A[0][0]**.
- *f(i,0)* = **A[*i*][0]** + 0.5 * **A[*i-1*][0]**.
- *f(0,j)* = **A[0][*j*]** + 1.5 * **A[0][*j-1*]**.
- *f(i,j)* = The minimum of (**A[*i*][*j*]** + 0.5 * **A[*i-1*][*j*]**) and (**A[*i*][*j*]** + 1.5 * **A[*i*][*j-1*]**).

Now, behold the following class definition:

```
typedef vector <double> VD;

class FindA {
  public:
     vector <VD> A;
     double f(int i, int j);
}
```

Implement **FindA::f()** as a recursive dynamic program with memoization. If you need to add a variable or two to the class, go ahead and do so. Your implementation should assume that the elements in **A** are already implemented, that each **A[i]** is the same size, and that *i* and *j* are less than **A.size()** and **A[0].size()** respectively.