

Name: _____

Math 203: Contemporary Mathematics

Chapter 6 test, version (a)

Tuesday, February 10, 2009

60 points

Instructions:

1. This test has 4 sheets of paper, 5 pages (counting the front and back of this sheet as two pages), and 2 questions. Please check that you have all of the pages.
 2. Read each question carefully. If you have any questions, please ask.
 3. Answer all of the following questions clearly and completely. Justify all of your answers. Most of the points you receive will be based on the accuracy, completeness, and clarity of your responses. Use full sentences, and avoid saying things that are untrue, ambiguous, or nonsensical.
 4. You may use a calculator for this test, but you may not use a book or any notes.
 5. Give your answer to each question completely and clearly in the space provided. You may use the back of the test pages for scratch work; however, if you want this work to be considered, please make note of it in the space provided for the question.
 6. Erase or cross out work you do not wish to be graded.
 7. You have 25 minutes to complete this test. Good luck!
-

Algorithms

Here are short descriptions of some of the algorithms that were explained in Chapter 6. You may find them useful. Note that we did not discuss all of these algorithms in class.

Breadth-first search, for finding the components of a graph:

1. Pick any vertex of the graph and highlight it.
2. Highlight all the edges connected to the highlighted vertex and all the vertices at the ends of those edges.
3. Repeat step 2 for all edges connected to any highlighted vertex.
4. When no more vertices can be highlighted, you are done and you have highlighted a connected part of the graph that is as large as possible; that is, you have highlighted a component.

(more algorithms on the back)

(another algorithm on the front)

Fleury's algorithm, for finding an Eulerian circuit or an Eulerian path:

1. Make a copy of the original graph and label it "Unnumbered Edges." Make a second copy of the vertices without the edges of the original graph and label it "Numbered Edges."
2. Choose any vertex of the original graph with unnumbered edges and highlight it as a selected vertex.
3. Consider all edges connected to the selected vertex. Remove one edge. Give it the next number (starting with 1) and shift it to the graph with numbered edges. Do not choose an edge that leaves behind a disconnected graph (that is, do not remove a bridge), unless the only edge attached to the selected vertex is a bridge. We give the shifted edge a number to keep track of the order in which the path is being constructed.
4. If the edge you removed was the last remaining edge in the whole graph with unnumbered edges, go to step 6.
5. If the edge you removed was not the last remaining edge, highlight the vertex on the other end of the removed edge as the new selected vertex. Now repeat step 3.
6. The numbered edges describe an Eulerian path.

Kruskal's algorithm, for finding a minimal spanning tree in a weighted graph:

As described in the textbook:

1. Consider only the vertices of the weighted graph.
2. Select the edge with the smallest weight and add that to the subgraph.
3. Consider the acceptable edges and choose the edge with the smallest weight. Add that to the subgraph.
4. Determine whether all vertices are connected by a path. If so, you have a minimal spanning tree. If not, repeat step 3.

A slightly different description of Kruskal's algorithm:

1. Begin your drawing of the minimal spanning tree by drawing only the vertices of the weighted graph.
2. Find the edge with the smallest weight, and draw that edge in your subgraph.
3. Find the edge with the next smallest weight, and draw that edge in your subgraph, **unless** that edge would complete a circuit in your subgraph.
4. Is your subgraph a connected graph? If so, you have a minimal spanning tree. If not, repeat step 3.

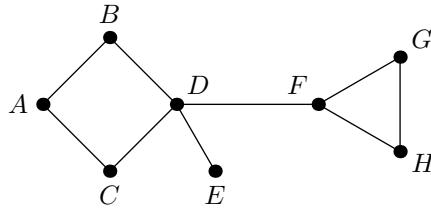
Nearest-neighbor algorithm, for finding a low-cost Hamiltonian circuit in a weighted graph:

1. Specify a starting vertex.
2. If unvisited vertices remain, go from the current vertex to the unused vertex that gives the least-cost connecting edge.
3. If no unvisited vertex remains, return to the starting vertex to finish forming the low-cost Hamiltonian circuit.

Cheapest-link algorithm, for finding a low-cost Hamiltonian circuit in a weighted graph:

1. In the beginning, all edges are acceptable and no edges have been selected.
2. From the set of acceptable edges, select the edge of smallest weight. If there is a tie, select any of the edges with the smallest weight.
3. If the selected edges do not form a Hamiltonian circuit, then determine the set of acceptable edges. Unacceptable edges are those that either share one vertex with two selected edges or that would close a circuit that is not a Hamiltonian circuit. Now repeat step 2.
4. If the selected edges form a Hamiltonian circuit, that circuit is your low-cost Hamiltonian circuit.

Question 1. (26 points) Answer the following questions about the graph below.



- (a) (5 points) How many vertices does this graph have? How many edges does it have?
- (b) (5 points) Is this graph connected or disconnected? Why?
- (c) (7 points) Is this graph a tree? Explain. If it is not a tree, can you find a subgraph which is a tree?
- (d) (9 points) List each vertex and its degree. Can you tell me something about this graph based on this information?

Question 2. (34 points) You are on vacation in Europe. The distances in kilometers between five European capitals, according to Google Maps, are given in the table below.

	Athens	Berne	Madrid	Prague	Vienna
Athens	—	2120	3290	1992	1708
Berne	2120	—	1546	823	865
Madrid	3290	1546	—	2329	2422
Prague	1992	823	2329	—	407
Vienna	1708	865	2422	407	—

(a) (10 points) Draw a weighted graph representing this information. Be sure to label the vertices of the graph.

(b) (8 points) You want to visit all five of these cities and return to where you started. In mathematical terms, what are you trying to find? Why?
 (Hint: Some of the terms we discussed in class are *component*, *Eulerian path*, *Eulerian circuit*, *subgraph*, *minimal spanning tree*, *Hamiltonian path*, *Hamiltonian circuit*, and *complete graph*. The answer to this question might be one of these.)

- (c) (16 points) You want to find a way to visit all five of these cities and return to where you started, but you would like to keep your total travel distance low. Use an algorithm to find an efficient way to tour these five cities. What is the total distance you travel if you go this way? (Be sure to state which algorithm you're using, and show all of your steps. Please feel free to use the back of this page or a sheet of scratch paper if you need more room.)

Name: _____

Math 203: Contemporary Mathematics

Chapter 6 test, version (b)

Tuesday, February 10, 2009

60 points

Instructions:

1. This test has 4 sheets of paper, 5 pages (counting the front and back of this sheet as two pages), and 2 questions. Please check that you have all of the pages.
 2. Read each question carefully. If you have any questions, please ask.
 3. Answer all of the following questions clearly and completely. Justify all of your answers. Most of the points you receive will be based on the accuracy, completeness, and clarity of your responses. Use full sentences, and avoid saying things that are untrue, ambiguous, or nonsensical.
 4. You may use a calculator for this test, but you may not use a book or any notes.
 5. Give your answer to each question completely and clearly in the space provided. You may use the back of the test pages for scratch work; however, if you want this work to be considered, please make note of it in the space provided for the question.
 6. Erase or cross out work you do not wish to be graded.
 7. You have 25 minutes to complete this test. Good luck!
-

Algorithms

Here are short descriptions of some of the algorithms that were explained in Chapter 6. You may find them useful. Note that we did not discuss all of these algorithms in class.

Breadth-first search, for finding the components of a graph:

1. Pick any vertex of the graph and highlight it.
2. Highlight all the edges connected to the highlighted vertex and all the vertices at the ends of those edges.
3. Repeat step 2 for all edges connected to any highlighted vertex.
4. When no more vertices can be highlighted, you are done and you have highlighted a connected part of the graph that is as large as possible; that is, you have highlighted a component.

(more algorithms on the back)

(another algorithm on the front)

Fleury's algorithm, for finding an Eulerian circuit or an Eulerian path:

1. Make a copy of the original graph and label it "Unnumbered Edges." Make a second copy of the vertices without the edges of the original graph and label it "Numbered Edges."
2. Choose any vertex of the original graph with unnumbered edges and highlight it as a selected vertex.
3. Consider all edges connected to the selected vertex. Remove one edge. Give it the next number (starting with 1) and shift it to the graph with numbered edges. Do not choose an edge that leaves behind a disconnected graph (that is, do not remove a bridge), unless the only edge attached to the selected vertex is a bridge. We give the shifted edge a number to keep track of the order in which the path is being constructed.
4. If the edge you removed was the last remaining edge in the whole graph with unnumbered edges, go to step 6.
5. If the edge you removed was not the last remaining edge, highlight the vertex on the other end of the removed edge as the new selected vertex. Now repeat step 3.
6. The numbered edges describe an Eulerian path.

Kruskal's algorithm, for finding a minimal spanning tree in a weighted graph:

As described in the textbook:

1. Consider only the vertices of the weighted graph.
2. Select the edge with the smallest weight and add that to the subgraph.
3. Consider the acceptable edges and choose the edge with the smallest weight. Add that to the subgraph.
4. Determine whether all vertices are connected by a path. If so, you have a minimal spanning tree. If not, repeat step 3.

A slightly different description of Kruskal's algorithm:

1. Begin your drawing of the minimal spanning tree by drawing only the vertices of the weighted graph.
2. Find the edge with the smallest weight, and draw that edge in your subgraph.
3. Find the edge with the next smallest weight, and draw that edge in your subgraph, **unless** that edge would complete a circuit in your subgraph.
4. Is your subgraph a connected graph? If so, you have a minimal spanning tree. If not, repeat step 3.

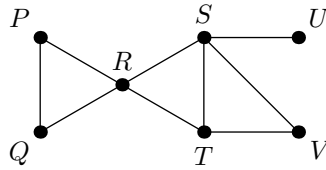
Nearest-neighbor algorithm, for finding a low-cost Hamiltonian circuit in a weighted graph:

1. Specify a starting vertex.
2. If unvisited vertices remain, go from the current vertex to the unused vertex that gives the least-cost connecting edge.
3. If no unvisited vertex remains, return to the starting vertex to finish forming the low-cost Hamiltonian circuit.

Cheapest-link algorithm, for finding a low-cost Hamiltonian circuit in a weighted graph:

1. In the beginning, all edges are acceptable and no edges have been selected.
2. From the set of acceptable edges, select the edge of smallest weight. If there is a tie, select any of the edges with the smallest weight.
3. If the selected edges do not form a Hamiltonian circuit, then determine the set of acceptable edges. Unacceptable edges are those that either share one vertex with two selected edges or that would close a circuit that is not a Hamiltonian circuit. Now repeat step 2.
4. If the selected edges form a Hamiltonian circuit, that circuit is your low-cost Hamiltonian circuit.

Question 1. (26 points) Answer the following questions about the graph below.



- (a) (5 points) How many vertices does this graph have? How many edges does it have?
- (b) (5 points) Is this graph connected or disconnected? Why?
- (c) (7 points) Is this graph a tree? Explain. If it is not a tree, can you find a subgraph which is a tree?
- (d) (9 points) List each vertex and its degree. Can you tell me something about this graph based on this information?

Question 2. (34 points) You are on vacation in Europe. The distances in kilometers between five European capitals, according to Google Maps, are given in the table below.

	Berlin	Budapest	Paris	Rome	Warsaw
Berlin	—	884	1056	1508	590
Budapest	884	—	1485	1351	843
Paris	1056	1485	—	1437	1600
Rome	1508	1351	1437	—	1837
Warsaw	590	843	1600	1837	—

(a) (10 points) Draw a weighted graph representing this information. Be sure to label the vertices of the graph.

(b) (8 points) You want to visit all five of these cities and return to where you started. In mathematical terms, what are you trying to find? Why?
 (Hint: Some of the terms we discussed in class are *component*, *Eulerian path*, *Eulerian circuit*, *subgraph*, *minimal spanning tree*, *Hamiltonian path*, *Hamiltonian circuit*, and *complete graph*. The answer to this question might be one of these.)

- (c) (16 points) You want to find a way to visit all five of these cities and return to where you started, but you would like to keep your total travel distance low. Use an algorithm to find an efficient way to tour these five cities. What is the total distance you travel if you go this way? (Be sure to state which algorithm you're using, and show all of your steps. Please feel free to use the back of this page or a sheet of scratch paper if you need more room.)