CS 216 Final Exam

You MUST write your name and e-mail ID on EACH page and bubble in your userid at the bottom of EACH page – including this page.

If you are still writing when "pens down" is called, your exam will be ripped up and not graded – even if you are still writing to fill in the bubble forms. So please do that first. Sorry to have to be strict on this.

Other than bubbling in your userid at the bottom, please do not write in the footer section of each page.

There are 12 pages to this exam – once the exam starts, please make sure you have all 12 pages.

Questions are worth different amount of points, from short answer questions worth 4 points each to long answer questions worth 12 points each. The short answer questions should not take more than a line or two to answer – *your answer should not exceed about 20 words.* There are 120 points of questions on the exam and 3 hours (180 minutes) to take the exam, which means you should spend about 1½ minutes per question point.

If you do not bubble in a page, you will not receive credit for that page! If the page is worth zero points, then you will receive a grade penalty.

This exam is CLOSED text book, closed-notes, **closed-calculator**, closed-cell phone, closed-computer, closed-neighbor, etc. Questions are worth different amounts, so be sure to look over all the questions and plan your time accordingly. Please sign the honor pledge here:

_____

_____

_____

_____

A crash reduces
Your expensive computer
To a simple stone.

(the bubble footer is automatically inserted in this space)

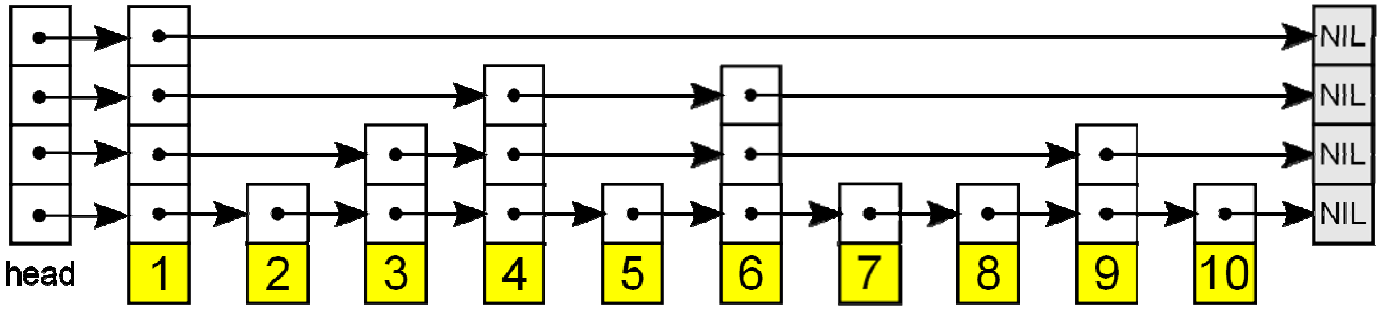
Exam 1 material

1. [6 points] Convert -6.5 to the hexadecimal notation for an IEEE 754 32-bit floating point number. Your answer should be in little endian format.

2. [6 points] Convert -53 to a 2's complement 32-bit integer.

Linked lists

Consider a new data structure that is similar to a linked list, but designed to be more efficient. A *skip list* is built in layers – the bottom layer is an ordinary ordered linked list. Each higher layer acts as an "express lane" for the lists below, where an element in layer *i* appears in layer *i*+1 with some fixed probability *p* (two commonly-used values for *p* are 1/2 or 1/4). On average, each element appears in 1/(1-*p*) lists.



A search for a target element begins at the head element in the top list, and proceeds horizontally until the current element is greater than or equal to the target. If the current element is equal to the target, it has been found. If the current element is greater than the target (or is the NULL pointer at the end of the list), the procedure is repeated after returning to the previous element and dropping down vertically to the next lower list.

3. [4 points] What is the big-theta running time of a search? Why? We are looking for worst-case here.

4. [4 points] What is the *expected* running time of a search? Why?

Exam 2 material

5. [4 points] What is the big-theta running time of a hash table search (assume a probing strategy of your choice for collision resolution, as they are generally the fastest executing collision resolution method).

6. [4 points] We have studied the C calling convention when we studied x86. While the convention is carefully laid out, it turns out that not all functions produced by a compiler follow the convention. When might a compiler purposely not follow the convention? This happens with good compilers (g++, for example), so the answer that the compiler is buggy or poorly designed is not what we are looking for here.

7. [6 points] Show an AVL tree double rotation (either one). You must show the 'intermediate' step, i.e. the tree after one of the two rotations. And your initial AVL tree must be in need of a double rotation.

C++ stuff

8. [4 points] Why do templates exist in C++? Meaning, what do they accomplish that we would not be able to do otherwise?

9. [4 points] What is dynamic dispatch? How do you 'activate' it in C++?

10. [4 points] What are the three uses of the const keyword in C++?

11. [4 points] What are the three main differences between pointers and references in C++?

Heaps and Memory

12. [4 points] What is difference between a heap and a priority queue? They are not the same thing!

13. [4 points] The running time for the heap operations insert() and remove_min() is logarithmic time. For a balanced tree (such as red-black), it can do the same operations (and many others!) in the same running time, yet heaps are used for priority queues instead of balanced trees. Why? List 3 such reasons.

14. [4 points] What properties of memory accesses that allow caches to work? Briefly, what do those properties mean?

Huffman

15. [12 points] Consider the following string (and hexadecimal number) – you need to build the Huffman coding tree for this string: "0x0xf7a07fa0fa403a7f0f7a7a047afa73a0700a0f74fa730f0f7a003a70". For your reference, there are 10 f's, 15 0's, 13 a's, 12 7's, 3 4's, 4 3's, and 1 x – don't include the double quotes, of course. You must also show heap that you use!

Graphs

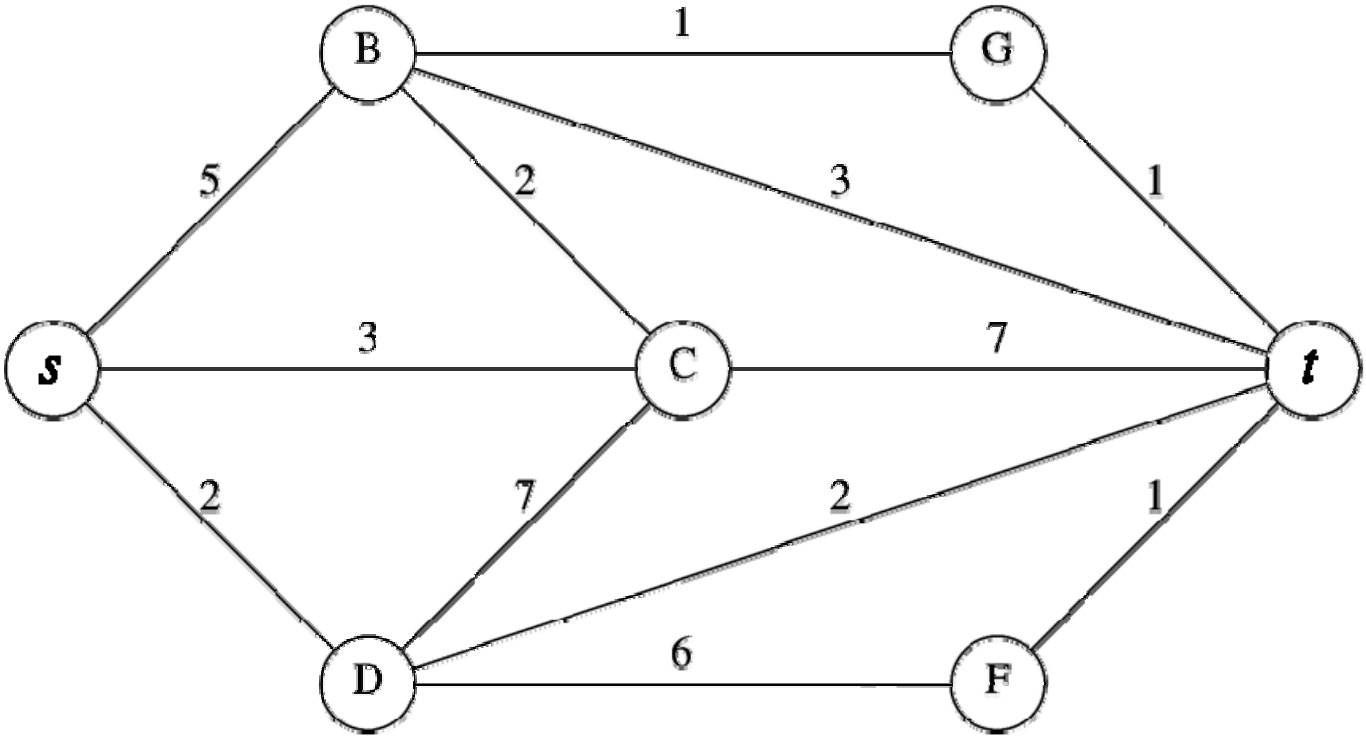
16. [4 points] BRIEFLY describe Dijkstra's shortest path algorithm

17. [4 points] How would you write a shortest path algorithm for a graph that has negative cost edges? You can assume that there are no negative cost cycles, only negative cost edges.

18. [4 points] What is the fundamental difference between Prim's algorithm and Kruskal's algorithm for finding a minimum spanning tree?

Graphs

Consider a new graph algorithm – finding the maximum flow through a graph. Given a source node *s* and a sink (or terminus) node *t*, we want to find the maximum flow that can leave *s* and arrive at *t*. Consider the graph used in the lecture set (the version below is not a directed graph):



Imagine that each edge was a pipe, with the capacity listed as the weight of each edge – in other words, the pipe from node A to node D can carry 2 gallons (or whatever unit) of water per time period (minute, etc). What, then, is the maximum amount of water that can leave the source node *s* and end at terminus node *t*? In this example, it is 10 (all 10 leave source node *s*; the 2 that go to D proceed directly to *t*; the 5 that go to B split up, with 3 going directly to *t*, 1 going through G to D, and 1 going to C; the 4 that go to C (3 from *s* and 1 from B) proceed directly to *t*). Note that there are multiple solutions for maximum flow, but the result is still 10.

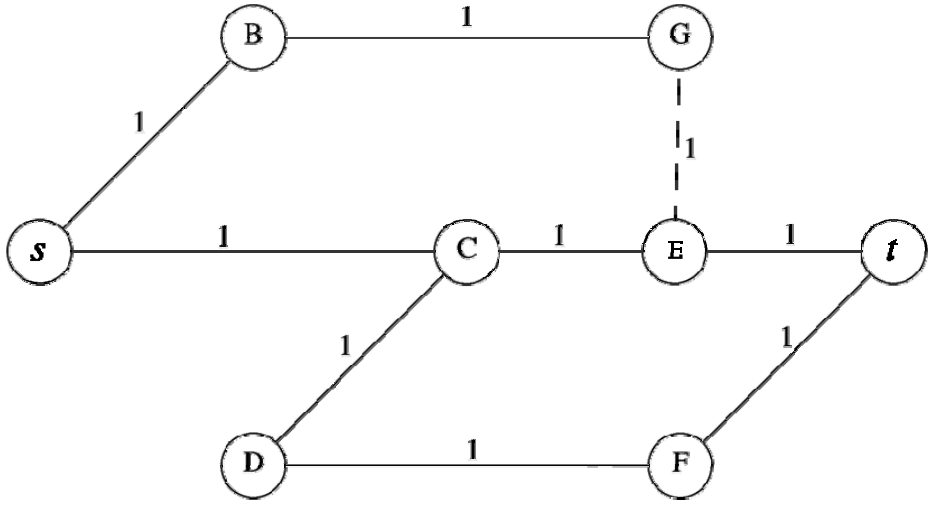
We first present an initial implementation of a maximum flow algorithm, which works as follows. Find the shortest path from *s* to *t* in the graph (here we are finding the shortest path by the number of edges being transverse, not the weight of the edges), and decrement the weight of each edge on the graph. If an edge ever hits zero weight, then that pipe is 'full', and no more flow can be sent along it. Repeat this process until there are no more paths between the source and the terminus (this will happen when enough edges hit zero capacity).

---

19. [4 points] What is the big-theta running time of the algorithm given above? Why?

20. [4 points] How might we optimize the algorithm to make it run faster?

21. [4 points] What is the big-Oh running time of the optimized algorithm (if different)? If it is the same, did the *expected* running time change? In either case, how (and why) did it change?



Consider the graph shown to the right. The algorithm as described above breaks down. The first path found is straight left to right: $s \rightarrow C \rightarrow E \rightarrow t$. After this first step, the three edges transversed have (remaining) capacity zero. However, there is another path that will increase the maximum flow through the graph: $s \rightarrow B \rightarrow G \rightarrow E \rightarrow C \rightarrow D \rightarrow F \rightarrow t$. This works because we are *decreasing* the flow from C to E – we'll call this *backflow*. The end result is that there is *no* flow along that edge. Thus, the final two paths of flow (both of capacity 1) are $s \rightarrow B \rightarrow G \rightarrow E \rightarrow t$, and $s \rightarrow C \rightarrow D \rightarrow F \rightarrow t$. The algorithm as described above cannot handle backflow, as it will not consider the edge between C and E after the first step, since its capacity was lowered from 1 to 0.

23. [8 points] How could we modify the algorithm to handle the problem of allowing backflow? This is not an easy problem, so don't spend too long on it!

UNIX

24. [4 points] Name all the g++ flags that you can remember, and BRIEFLY describe what they do.

25. [4 points] List three advantages of UNIX over Microsoft Windows (or UNIX over Mac OSX).

26. [4 points] List three advantages of Microsoft Windows over UNIX (or Mac OSX over UNIX).
