COS 226
Final Exam Review
Fall 2015

Ananda **Guna**wardena

(guna)

guna@cs.princeton.edu

Logistics

- The final exam **time and location**
  - 7:30-10:30 PM on Friday, January 15. McCosh 50
  - The exam will start and end promptly, so please do arrive on time.
- **Exam Format**
  - Closed book, closed note.
  - You may bring one 8.5-by-11 sheet (both sides) with notes in your own handwriting to the exam.
  - No electronic devices (e.g., calculators, laptops, and cell phones).

What to focus on

- focus on understanding fundamentals, not memorizing details (eg: code)
- ***For each algorithm***
  - *Write down as many algorithms as you can recall*

KMP, Burrows, Boyer-Moore, BFS, DFS, MSD, LSD, 3-way qsort, Rabin-Karp, Kruskal, Prim, Dijkstra, Ford-Fulkerson, Huffman, LZW, run-length encoding,

  - understand how it works on typical, worst case, best case input

Consider algorithm class (eg; compression) and think of inputs that give various performances

  - How is it different from other algorithms for the same problem?

Compression – Huffman, LZW, run-length, Burrows-Wheeler
String Search – KMP, Boyer-Moore, Rabin-Karp
String Sort - LSD, MSD, 3-way quicksort
Max Flow – Ford-Fulkerson

What to focus on

- ***For each data structure***
  - *Write down as many data structures as you can recall*

R-way Tries, TST, Hash Tables, Graphs, diGraphs, Priority Queue, ST

  - Why do we care about this data structure?

Each data structure is for a specific purpose. Like Tries are for storing strings so we can do efficient substring search

  - What makes us choose one data structure over another?

The goal of choosing a specific data structure is to make the operations more efficient. For example, we can use an array to store any data. But it will not be efficient to search, delete etc

Material covered

- The exam will *stress* material covered since the midterm, including the following components.
  - Lectures 13–23.
  - *Algorithms in Java, 4th edition*, Chapters 4–6.
  - Exercises 12–22.
  - Programming assignments 6–8
    - Wordnet, seam-carving, burrows-wheeler

Topics covered

Depth-first search	Breadth-first search
Kruskal's algorithm	Dijkstra's algorithm
Key-indexed counting	LSD radix sort
Knuth-Morris-Pratt substring search	Boyer-Moore substring search
RE to NFA	R-way tries
Run-length coding	Huffman coding
Topological sort	Prim's algorithm
Bellman-Ford algorithm	Ford-Fulkerson algorithm
MSD radix sort	3-way radix quicksort
Rabin-Karp substring search	
Ternary search tries	Reductions
LZW compression	Burrows-Wheeler

Wordnet

1. What data structures are used to store wordnet?
   DiGraph, HashTables
2. What data structures are used to store SCA?
   Digraph
3. Is there a reason that we used BFS not DFS?
   DFS would not work. See precept notes for an example. You always want to use BFS for shortest path algorithms
4. What is the order of the best algorithm that can find the length of the common ancestor?
   V + E (require some sort BFS)
5. What is the order of the best algorithm that can find the common ancestor?
   Same as (4)
6. What is a rooted DAG and how do we determine that? Order or growth of your algorithm?
   There exits a single node whose outdegree is 0 and graph is directed and acyclic. Can find the outdegree of all nodes in V + E (just follow the adjacency list) and can determine graph has no cycles with DFS
7. If the wordnet is NOT a rooted DAG, will answers to 3 and 4 will hold?
   No. There may not be a SCA if this is the case (nodes in two disjoint subtrees)
8. Given a list of n nouns, What is the order of growth of the outcast algorithm?
   N^2 (V + E) - There are n^2 pairs that needs to be checked

Seam-Carving

- What is the purpose of the seamcarving assignment?
  - To resize an image w/o skewing the figures
- How does it relate to shortest path?
  - The problem can be viewed as finding the shortest path of a DAG. There is a V + E algorithm for doing this. One question is how many nodes this graph has? (n^2 vertices and each vertex a max possible 3 outgoing edges. And so n^2 + 3 n^2 = 4 n^2
- How to find Vertical and Horizontal seams?
  - Find the pixels with lowest energy total and remove
- Why do a defensive copy of Picture?
  - To reuse the original copy for many removeSeam operations
- What is the order of growth for the two methods, removeHorizontalSeam and removeVerticalSeam?
  - H * W
- Can seamcarver be extended to video scaling? How?
  - Possible since each video is a collection of images, in theory this can be done. However, there are many details to be worked out and it is possible there are better algorithms to do this

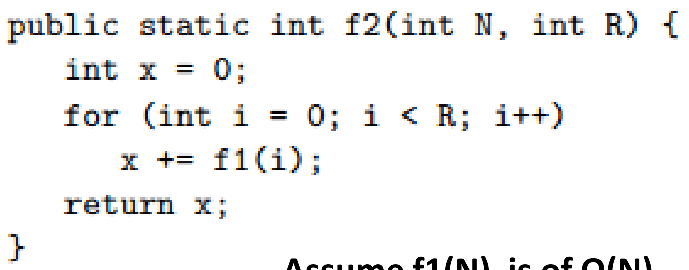
Burrows-Wheeler

- What libraries were used to read and write input/output to the program?
  - BinaryStdIn and BinaryStdOut
- What method in the output library that was required to print the output correctly?
  - HexDump, close and flush
- What data structures were used to implement BW, CSA, MoveToFront?
  - BW – uses a CSA (circular suffix array)
  - SCA – uses a Array and MoveToFront uses an ARRAY
- What is the order of growth to form circular suffixes of a given string?
  - Can be done in linear time. One trick is to form a new string S + S and then consider suffixes just using two things – Reference to starting character, length of the suffix
- What is wrong with using LSD.sort() in the program?
  - Quadratic in performance
- Could we have used quicksort to sort suffixes? How? If so how can you avoid quadratic performance?
  - We could have used quicksort, however, the compareTo should be carefully designed to avoid a suffix comparing to itself (can lead to quadratic performance)

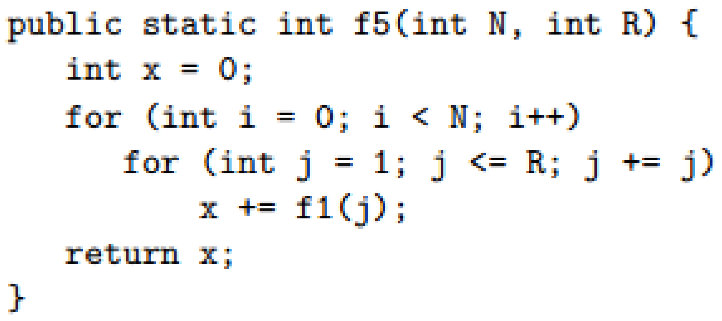
Burrows-Wheeler ctd…

- What are the 3-steps to burrows-wheeler transform?
  - BW ➔ MoveToFront ➔ Huffman encoding
- How would you sort strings w/o forming them explicitly?
  - You can create suffixes just using a reference to first character and length of the suffix
- Is it necessary to do move to front? If not, why did we do it?
  - You don't need to, but w/o it would not have any benefit to BW. It create many duplicates which is good for huffman
- How did we do the inverse transform?
  - We had to use a **next** array

Analysis of Algorithms



```
public static int f2(int N, int R) {
   int x = 0;
   for (int i = 0; i < R; i++)
      x += f1(i);
   return x;
}
```



```
public static int f5(int N, int R) {
   int x = 0;
   for (int i = 0; i < N; i++)
      for (int j = 1; j <= R; j += j)
         x += f1(j);
   return x;
}
```

Assume f1(N) is of O(N)

$f2(N, R) = \sum f1(i) \quad i=0..R$

$= \sum i = O(R^2)$

$f5(N, R) = N(1 + 2 + 4 + ... + R)$

$= N(1 + 2 + 4 + ... + R)$

$\sim N(2R) = O(NR)$

Analysis of Algorithms ctd..

Assume f1 is of O(N)

```
public static int f4(int N) {
   if (N == 0) return 0;
   return f4(N/2) + f1(N) + f1(N) + f1(N) + f4(N/2);
}
```

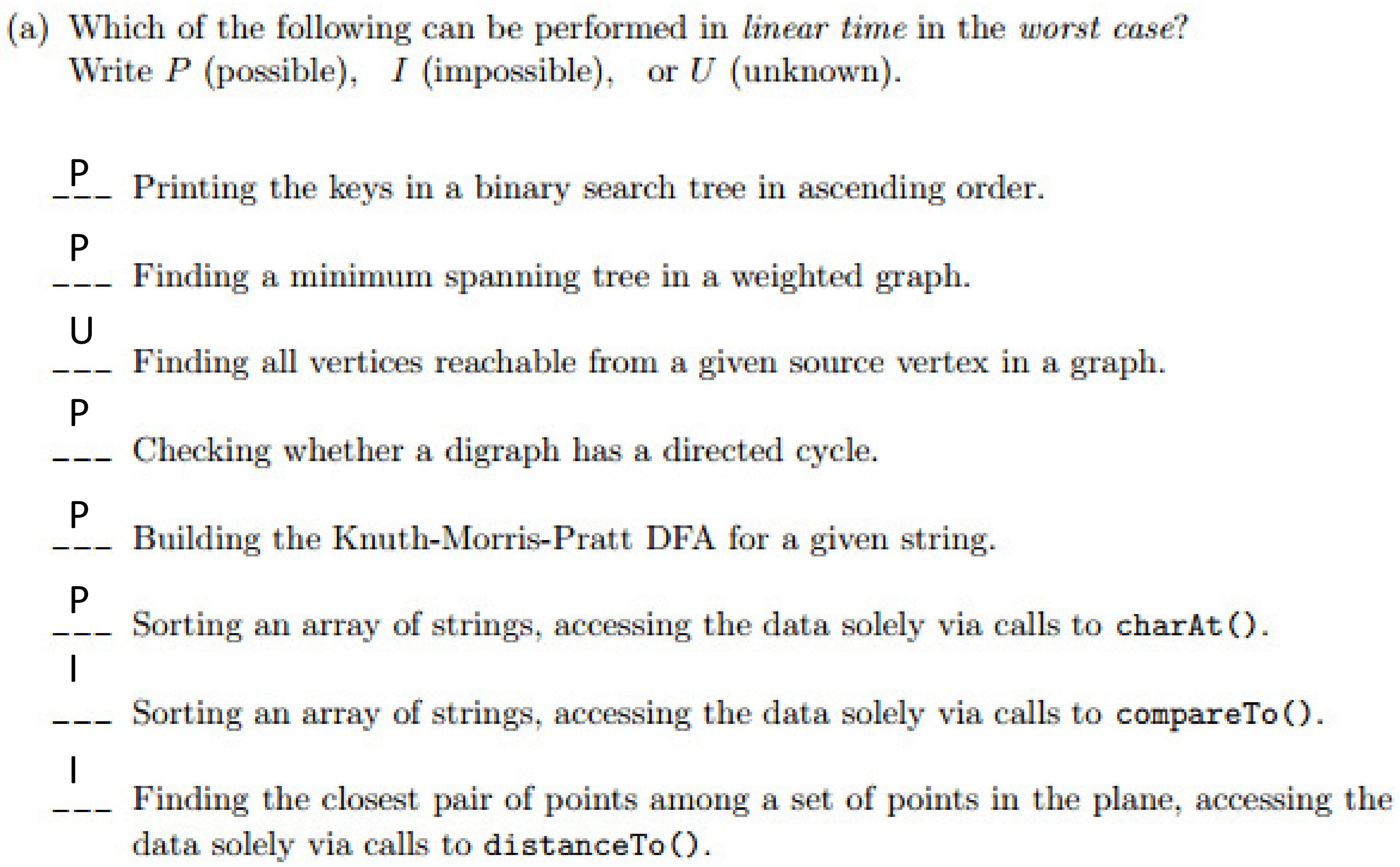
```
public static int f3(int N) {
   if (N == 0) return 1;
   int x = 0;
   for (int i = 0; i < N; i++)
      x += f3(N-1);
   return x;
}
```

f4(N) = 2 f4(N/2) + 3 f1(N)
= 2(2f4(N/4) + 3 f1(N) + 3 f1(N)
= 2^2 * f4(N/2^2) + 2* 3N

= 2^k * f4(N/2^k) + k* 3N
Assume N = 2^k
= N * f4(1) + log N * 3N
~ O(N log N)

f3(N) = N * f3(N-1)
f3(N) = N!

Classifying Algorithms



(a) Which of the following can be performed in *linear time* in the *worst case*? Write *P* (possible), *I* (impossible), or *U* (unknown).

P ___ Printing the keys in a binary search tree in ascending order.

P ___ Finding a minimum spanning tree in a weighted graph.

U ___ Finding all vertices reachable from a given source vertex in a graph.

P ___ Checking whether a digraph has a directed cycle.

P ___ Building the Knuth-Morris-Pratt DFA for a given string.

P ___ Sorting an array of strings, accessing the data solely via calls to `charAt()`.

I ___ Sorting an array of strings, accessing the data solely via calls to `compareTo()`.

I ___ Finding the closest pair of points among a set of points in the plane, accessing the data solely via calls to `distanceTo()`.

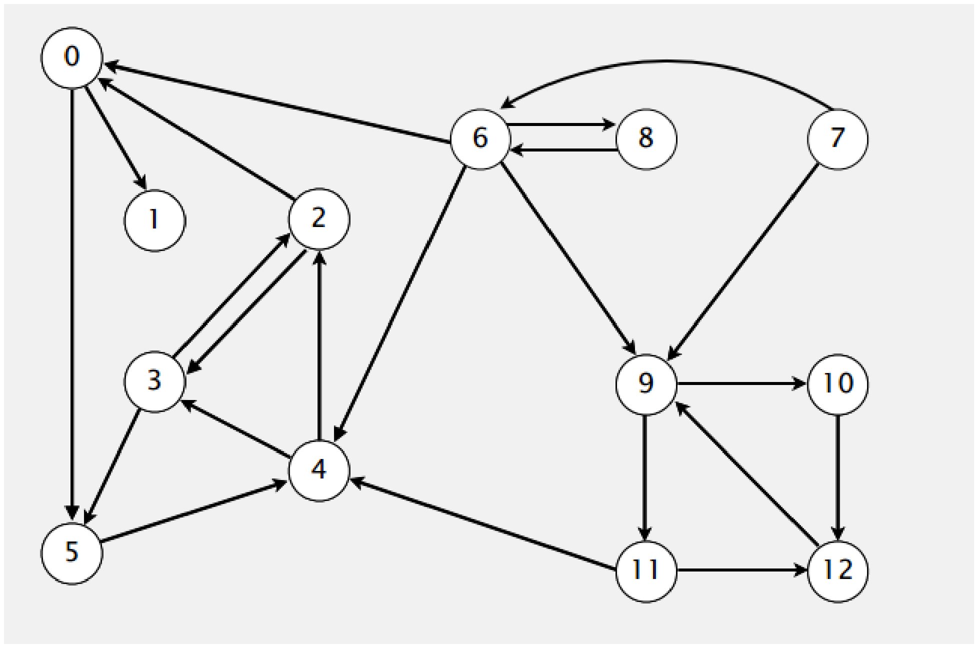
Mystery Code

```
for (Edge e : G.adj(v))
{
   int w = e.to();
   if (dist[w] > dist[v] + e.weight())
   {
      dist[w] = dist[v] + e.weight();
      pred[w] = e;
      pq.insert(dist[w], w);
   }
}
```

This code belongs to one of the graph algorithms. Which one(s)?

This is a piece of code related to weighted DiGraphs.
This is from Dijkstra's shortest path algorithms

Finding SCC's



What is the difference between **connected components** and **strongly connected components**?

Steps of Kosaraju-Sharir

1. **Find the reverse graph G^R**
2. **Find the reverse post order of the reverse graph G^R**
3. **Traverse the graph G in the order of 2. All vertices in the same DFS will have the same components number**

Hashing

- When implementing a ST with hashing, what operations are not allowed in the ST?
  - **HashTable does not support order operations and hence ST cannot support any order operations (eg: rank)**
- What is a collision and how do we avoid them?
  - **When two entries k1 and k2 hash to the same location or H(k1) = H(k2). You cannot completely eliminate them unless you have a perfect hash function**
- How can we minimize collisions in a hashtable?
  - **Choose a uniform hash function**

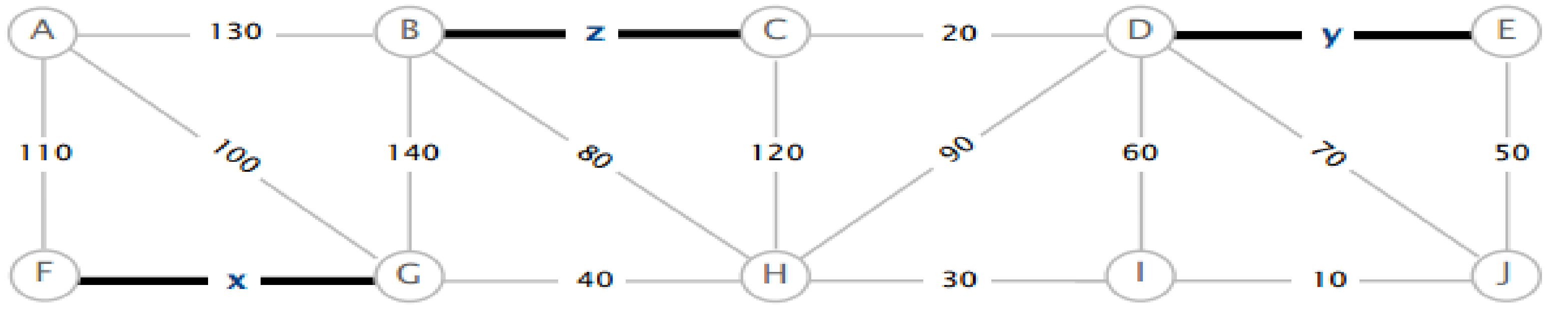
Hashing

- True/False
  - A linear probing hash table always finds a place
    - **True if the number of keys <= table size**
  - A quadratic probing hash table always find a place
    - **Not discussed in lecture**
  - A separate chaining hash table always finds a place
    - **True with no condition on table size and keys**
  - The load factor of a hash table is always <= 1
    - **True is linear probing table. Not necessarily true if separate chaining is used**
  - A linear probing hash table must be rehashed if load factor is over 0.7
    - **If the table is 70% full it is a good idea to rehash to avoid clusters that can lead to poor performance**
  - A rehashed entry will be at the same location as the original
    - **Usually we double the size of the table (amortized constant time) and because table size changes, it may not be the same place. For example H(s) % N and H(s) % 2N are not the same. Assume H(s) = 25 and N = 15**

Hashing question

- Suppose 10,000 strings of length 5 from ASCII table is hashed using the hash function
  - H(S) = ∑ s[i]   I =0…4 (table size = 10,000)
- Questions
  - Is this a good hash function? Why or why not?
    - **No. Since all permutations of s will lead to the same value and there are n! permutations or length n strings**
  - If linear probing hash table is used, what is the probability of a collision (after hash table has 1280 elements)?
    - **Since 0 <= H(s) < = 256*8 = 1280, after 1280 elements are written all other entries will hash to the same place. So the probability of a collision is (10000-1280)/10000**
  - If separate chaining is used, what is the average length of a chain?
    - **In this case, each chain will be 10000/1250 = 8 (assuming uniform hashing)**

MST

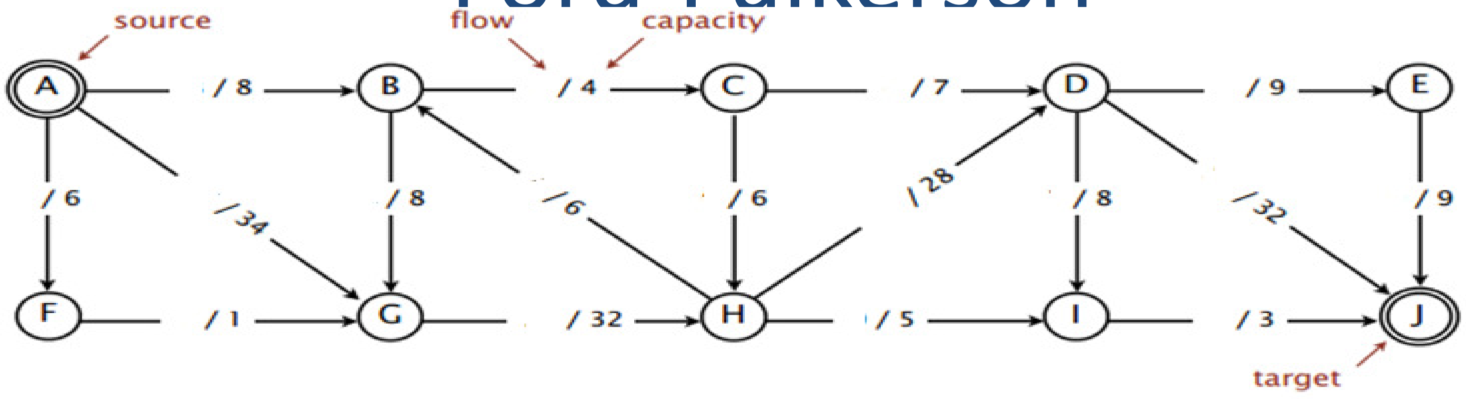


If edges X, Y and Z are in the MST

1. Find the other edges that are in MST
    - **There are 10 nodes in the graph and hence we need 9 edges in the MST**
    - **Since X, Y, Z are included, we need to find the smallest set of 6 other edges that does not form a cycle.**
    - **We know : 10, 20, 30, 40, 50, and 110 are the next ones**

2. find upper bounds for edge costs of X, Y and Z?
    - **X <= 110 (otherwise we can replace X with A-F**
    - **Y <= 70 (otherwise we can replace Y with D-J**
    - **Z <= 80 (otherwise we can replace Z with B-H**

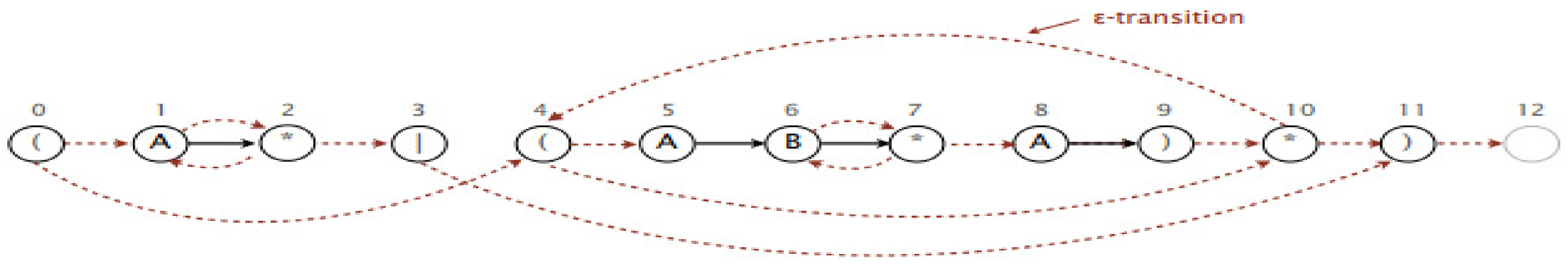
Ford-Fulkerson



1. What is the possible max flow of the network?
    - This is sum of all flows into j = 32 + 9 + 3 = 44
2. Mark an augmenting path and increase flow
    - Pick any path with no full forward edges and no empty backward edges (A – G – H – I – J). Bottle neck capacity is 3
3. Find all augmenting paths and increase flow
    - Do this as a homework. Do this as long as there are augmenting paths
4. What is the actual max flow?
    - Actual max flow can be calculated when there are no more augmenting paths
5. What is a min-cut?

   A min-cut is a cut such that forward edges from s-cut to t-cut are full and backward edges are empty
6. Min-cut can only be calculated when a certain condition is true. What is it? How do we find out?
    - No more augmenting paths.

Regular Expressions and NFA



1. What is the regular expression? **(A * | (A B * A) *)**
2. Suppose that you simulate the following sequence of characters on the NFA above: **A A A A A A A** . In which one or more states could the NFA be?
    - **Read null ➔ { 1,2,3,11,12,4,5,10,} - hence null char is accepted (state 12)**
    - **Read A ➔ {1,2,3, 11, 12, 6, 7, 8, 10} - accept A**
    - **Read AA ➔ { 1,2,3.11,12, 9, 10} - accept AA**
    - **Suppose we see if B is accepted**
        - **After reading null you can be in states { 1,2,3,11,12,4,5,10,}**
        - **After reading B next, there is no transition that you can take. Hence B is not accepted**
3. Suppose that you want to construct an NFA for the regular expression (A * | (A B * A) +) where the operator + means one or more copies. What minimal change(s) would you make to the NFA above?

Compression Algorithms
Run-length encoding

- Describe the algorithm
  - **Encode runs of 0's and 1's. Say you have 15 0's followed by 25 1's. You can just represent this as 15 25**
  - **Since 15 and 25 can be represented using 5 bits, we need only 10 bits to represent a total of 15+25 = 40 bits. A compression ratio of 10/40 = 25%**
- Under what circumstances would you use this algorithm?
  - **A great algorithm when sending FAX, lots of white spaces (0's) leading to long patterns of 0**
- If 8-bit words are used to store counts, what is the length of the maximum run that can be stored?
  - **You can store up to 255 bits (max number represented with 8-bits) in 8-bit code word**.
- What can we do if the length of the run cannot be accommodated by n-bit word?
  - **Use a dummy 0. For example, if you have 1000 , 0's then we can use 255 0 255 0 255 etc**
  - **The 0 will not produce anything when decoding**
- What is the best case input for run-length encoding (8-bit code words)
  - **Every 255 bits is represented by just 8-bits. 8/255 compression ratio**

Compression Algorithms
Huffman encoding

- Describe the algorithm
  - **Build a frequency table of characters. Use a huffman tree to find prefix-free code such that chars with higher frequencies have lower bits**
- What is the preprocessing step of the algorithm?
  - **Building the huffman code table. This code table must be the compressed file header and must be transitioned with the file**
- What data structure(s) is/are used in the preprocessing step?
  - Need a PQ for finding the two mins to build a huffman tree. We also need an array to keep counts
- If you compress a file with all characters the same (eg: 10000 A's) what is the compression ratio?
  - Since all chars are the same we can represent A by just 1 bit (0 or 1). So we can get 1:8 compression
- Describe a situation where no compression is obtained
  - **Suppose all 256 chars have the same frequency. Now the huffman tree will be log (256) = 8 in depth and each char now have to be encoded with b bits (this is same you need for uncompresed file). So the compression is 1:1 (no compression**)

Compression Algorithms
LZW encoding

- Describe the algorithm
  - The key here is to recognize that there are repeated patterns in a file (eg; novel repeated words) and they can be mapped to a number and use the number to compress a long pattern. For example, if we use 12-bits to represent a code word, there are 2^12 patterns that can be stored. Since we need 256 for the original alphabet, we can store 2^12 – 256 new patterns. Any thing more than that is probably useless as many long patterns may never be repeated.
- What data structure is used to store code words?
  - We can use a ST (key = pattern, value = 12-bit number that represent pattern)
- Is it possible to run out of code words to store new words? How?
  - Yes, after 2^12 – 256 new words we will run out. But we probably don't need anymore patterns anyway to get a good compression
- Should we send the code words with the compressed file?
  - No, there is no need. Besides can you imagine how costly it is to send the ST
- How can we decompress a file?
  - Same process, you need the alphabet to start and then you build the ST as you go
- What is the tricky case and how do we overcome that?
  - When a patterns such as : sW sW s (s = char, W = word) and note that as soon as sW is placed on the ST, it is immediately used, so we miss a number

LZW

97	**a**	
98	**b**	ab =128
128	**ab**	ba = 129
129	**ba**	abb = 130
131	In trouble the code word 131 is not in the ST yet . So use the ba(previous code) + b (first char of the prev code) = **bab** = 131	
132	Same here = **bab + b** = 132	
130	**abb**	

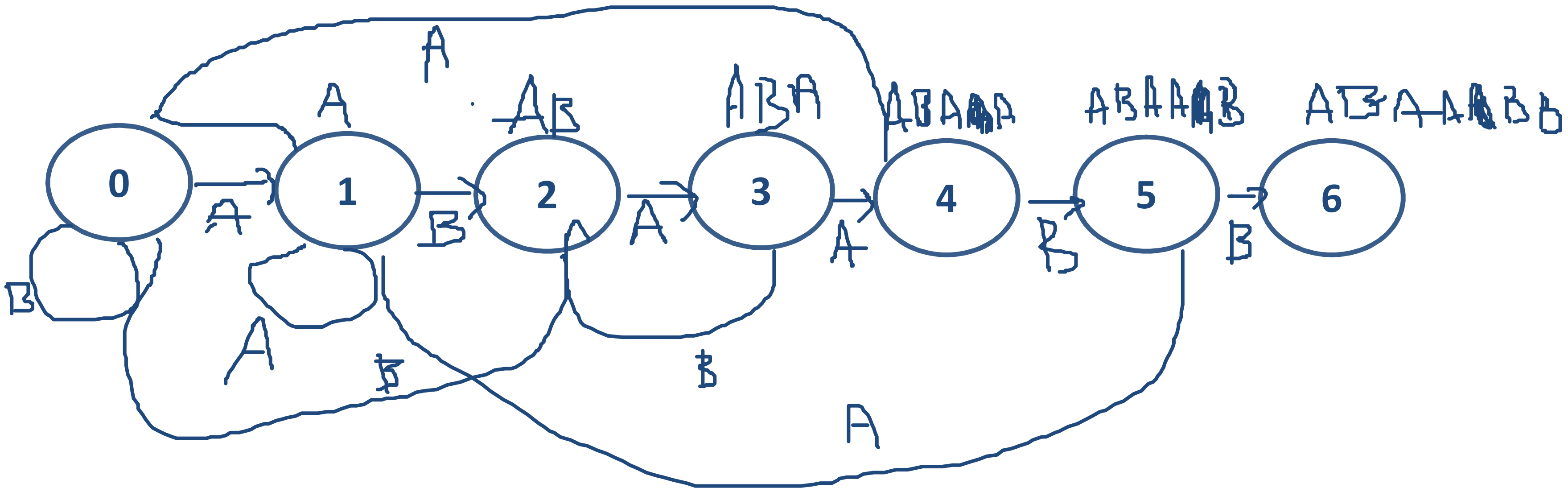
Decode the message a=97, b=98, and start next token from 128

KMP Algorithm

- Briefly describe the algorithm
  - **This is one of the genius algorithms. The key idea is not to go back with text pointer so we can do a pattern search with m(pattern) + n (text) – Note that brute force is m*n**
- What is the order of growth of building the DFA? Typical algorithm? Best algorithm?
  - **DFA can be built in m^2 time, but can do in linear time m (see notes)**
- What is the order of growth of the algorithm for searching for a pattern of length m in a text of length n?
  - **m + n**
- Can KMP be adjusted to find all occurrences of a pattern in a text? What is the order of growth?
  - **Sure, just reset the DFA and keep going. It is still the same = m + n**
- Is KMP the algorithm of choice for any substring search application?
  - **Not always there can be other algorithms that work better than m + n**

Exercise

- Build the DFA for : ABAAABB



Exercise

	0	1	2	3	4	5	6	7	8	9	10
A	0	0								10	11
B					5		2				4
s	B	B	A	B	B	B	A	**B**	B	A	A

Complete the table and find the search string

At state 10, we saw a B and had to go back to 4. Since Table [B][2]= A Table[B][3] = B

At state 6, we saw a B and had to go back to 2. Since Table [B][1]= B this must be a B

Boyer Moore (BM)

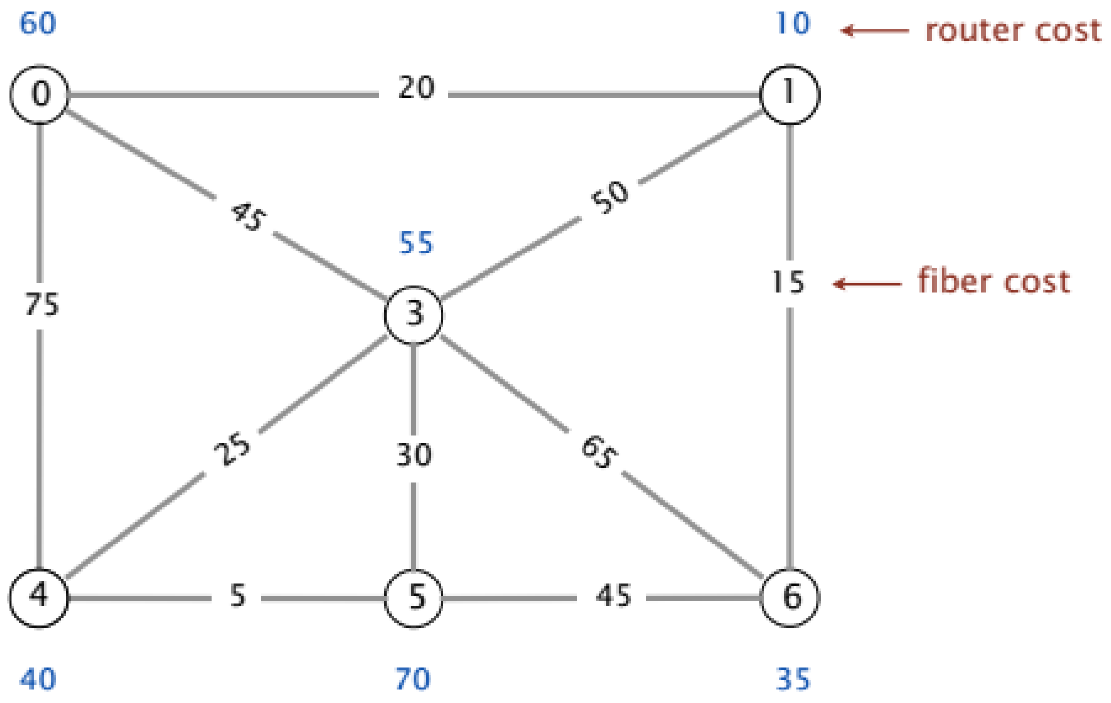
- Briefly describe the algorithm
  - **Instead of left to right, start comparing pattern and text from right to left. So we will find mismatches soon and can make bigger jumps.**
- What is the pre-processing step of Boyer-Moore?
  - **Just creating the table of how much to jump when a mismatch occur**
- What is the order of growth of the algorithm for searching for a pattern of length m in a text of length n? Best case? Worst case?
  - **Best case is after one comparison we jump the pattern length. So we have n / m (sub linear). Worst case is m*n (imagine searching BAAA in AAAAAAAAAAAAAAAABAAA**
- Can BM be adjusted to find all occurrences of a pattern in a text? What is the order of growth?
  - **Yes. Keep looking after finding the first pattern.**

Rabin-Karp (RK)

- Briefly describe the algorithm
- What is the pre-processing step of RK algorithm?
- What is the order of growth of the algorithm for searching for a pattern of length m in a text of length n? Best case? Worst case?
- Can RK be adjusted to find all occurrences of a pattern in a text? What is the order of growth?

Design Problem connecting dorm rooms

There are N dorm rooms, each of which needs a secure internet connection. It costs $w_i > 0$ dollars to install a secure router in dorm room i and it costs $c_{ij} > 0$ dollars to build a secure fiber connection between rooms i and j. A dorm room receives a secure internet connection if either there is a router installed there or there is some path of fiber connections between the dorm room and a dorm room with an installed router. The goal is to determine in which dorm rooms to install the secure routers and which pairs of dorm rooms to connect with fiber so as to minimize the total cost.



See design session solution

Bonus Material

Reduction

HAM-PATH. Given a digraph, is there a path that visits each vertex exactly once?

SHORTEST-SIMPLE-PATH. Given a digraph with integer edge weights (positive or negative), two distinguished vertices s and t, and an integer L, is there a *simple* path from s to t of length at most L. (A simple path is a path that visits each vertex at most once.)

Show that Ham-Path polynomial reduces to Shortest-Simple-Path.

Create a new weighted digraph G' as follows:

- G' has the same vertices as G plus two new vertices s and t.
- G' has the same edges as G plus a new edge from s to every vertex in G and an edge from every vertex in G to t.
- The weight of every edge is -1.

Observe that G has a Hamiltonian path if and only if G' has a shortest simple path from s to t of length exactly $-(V+1)$.

Reduction

ELEMENTDISTINCTNESS. Given N real numbers, are any two of them equal?

CLOSESTPAIR. Given N points in the plane, find a pair that is closest in Euclidean distance.

Show that ElementDistinctness linear reduces to ClosestPair

Given an instance $x_1, \ldots x_N$ of ELEMENTDISTINCTNESS, form the instance $(x_1, 0), \ldots, (x_N, 0)$ for CLOSESTPAIR. The elements in the ELEMENTDISTINCTNESS problem are distinct if and only if the closest pair of points has distance strictly greater than 0.

Remark. There is an $\Omega(N \log N)$ lower bound for ELEMENTDISTINCTNESS in the quadratic decision tree model of computation. This reduction proves that there is also an $\Omega(N \log N)$ lower bound for CLOSESTPAIR.

NFA

Draw an NFA that recognizes the same language that the regular expression a(bc)*d | e* describes. Use the notation and construction given in lecture.

KMP, Boyer-Moore, Rabin-Karp

- KMP – how does it work – examples

- Boyer-Moore – how does it work

- Rabin-Karp

Algorithm Analysis

1. Order of growth

```
public static int f3(int N) {
   if (N == 0) return 1;
   int x = 0;
   for (int i = 0; i < N; i++)
      x += f3(N-1);
   return x;
}
```

```
public static int f2(int N) {
   int x = 0;
   for (int i = 0; i < N; i++)
      for (int j = 0; j < i; j++)
         x++;
   return x;
}
```

2. Order of growth

Suppose that you collect the following memory usage data for a program as a function of the input size N.

N	memory
1,000	10,000 bytes
8,000	320,000 bytes
64,000	10,240,000 bytes
512,000	327,680,000 bytes

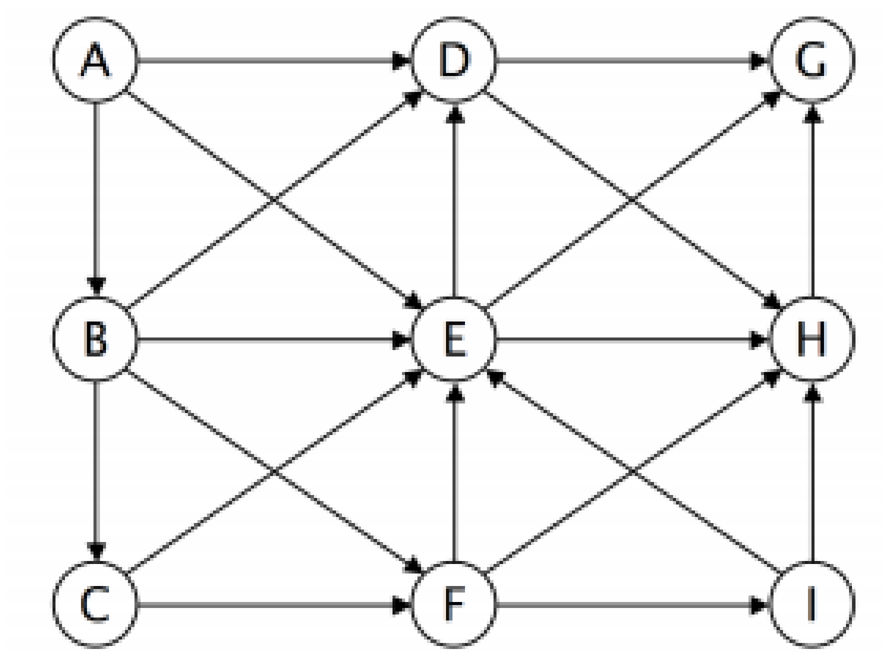
Estimate the memory usage of the program (in bytes) as a function of N and use tilde notation to simplify your answer.

Challenge Questions

- **Answer each question with possible, impossible, unknown**
  - It is possible that we may find an algorithm that can build a BST in linear time
  - There exist an algorithm where duplicity of elements in a set can be determined in sub-linear time
  - The convex hull problem (i.e. finding a set of points that encloses a given set of n points) can be solved in linearithmic time
  - Inserting n comparable keys into a BST in time proportional to n

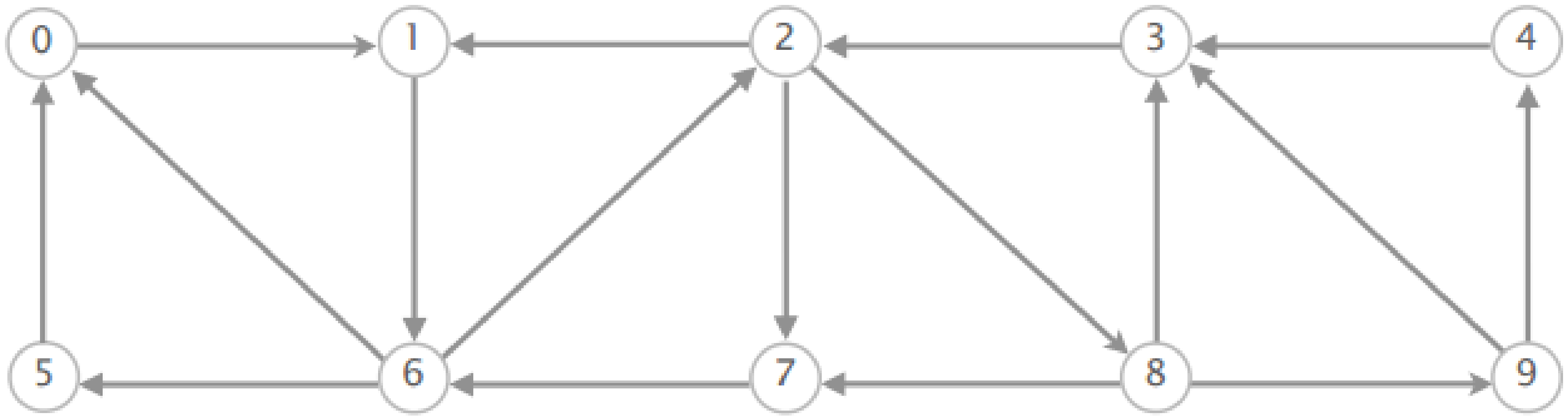
Graph Algorithms

3. Graph Search



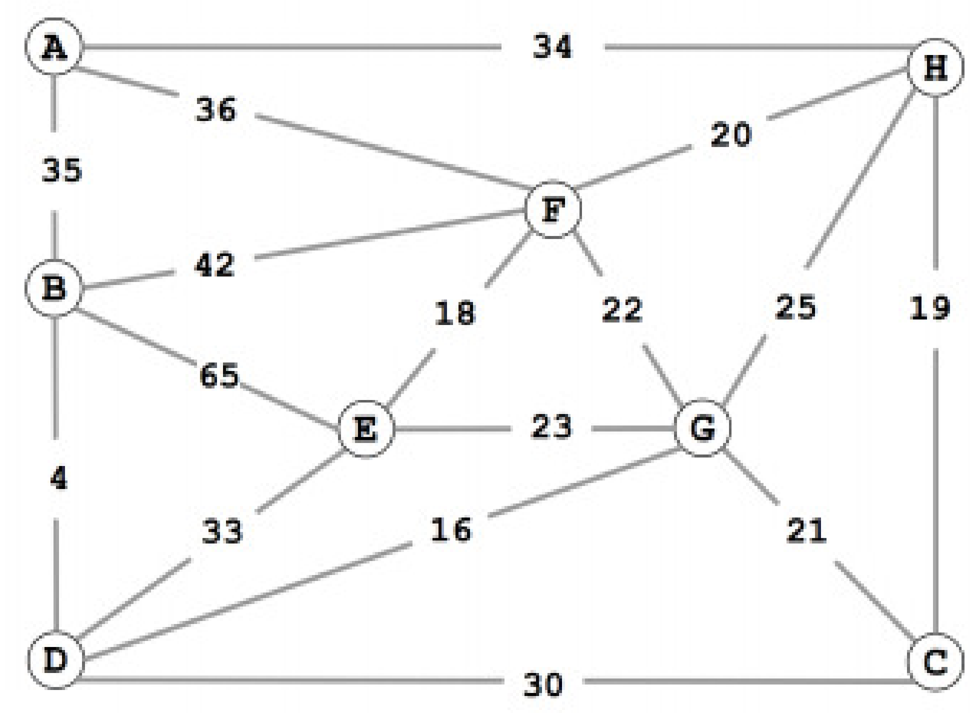
- Run BFS, DFS (starting from vertex A)
- Identify one situation where you would need to use BFS instead of DFS.
- Identify one situation where you would need to use DFS instead of BFS.

4. Order traversals



List the vertices in preorder, post order and reverse post order (starting with node 0)

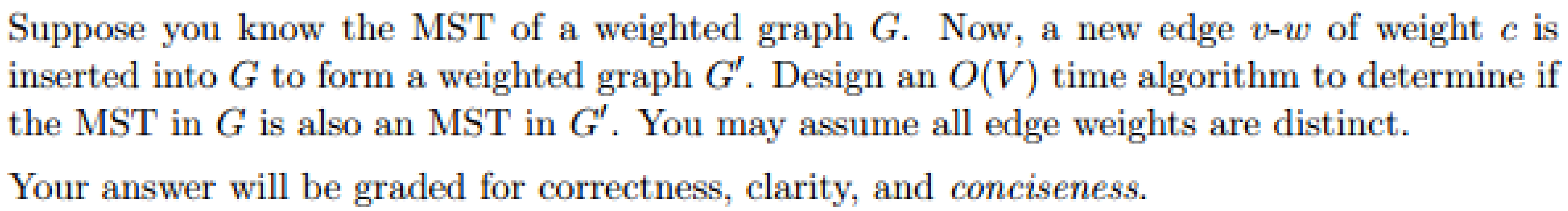
5. MST



- Run Kruskals

- Prims

6. MST Algorithm Design



Suppose you know the MST of a weighted graph G. Now, a new edge v-w of weight c is inserted into G to form a weighted graph G'. Design an $O(V)$ time algorithm to determine if the MST in G is also an MST in G'. You may assume all edge weights are distinct.

Your answer will be graded for correctness, clarity, and *conciseness*.

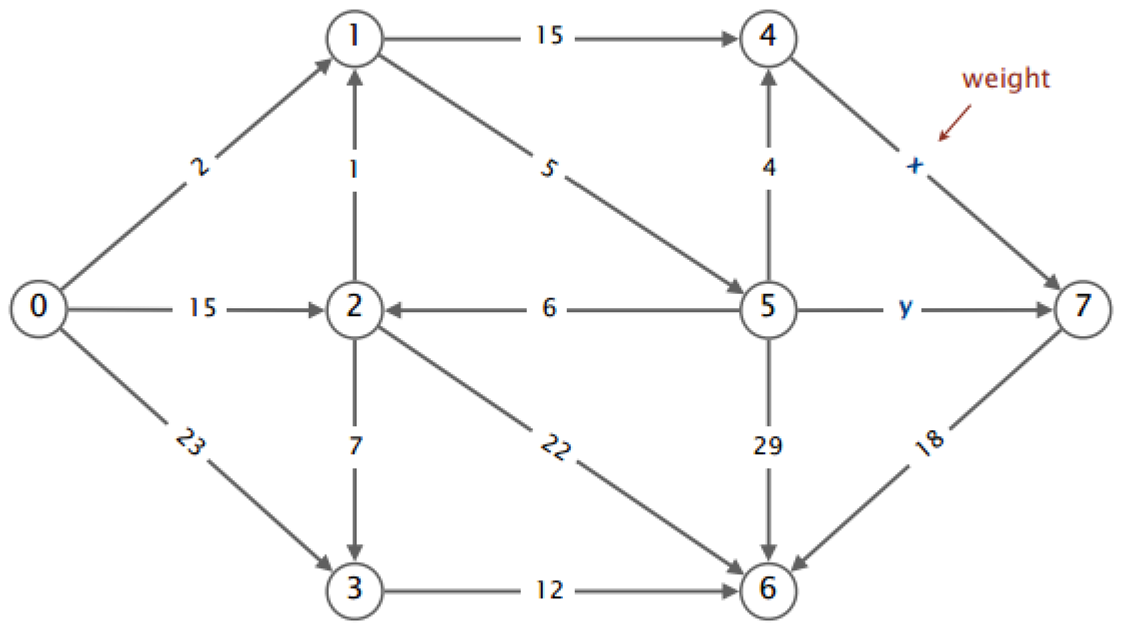
1. State the algorithm

2. Explain why your algorithm takes O(V) time

7. Match Algorithms

___ T9 texting in a cell phone	A. Trie
___ 1D range search	B. Hashing
___ 2D range search	C. 3-way radix quicksort
___ Document similarity	D. Binary search tree
___ Traveling salesperson problem	E. Kd tree
___ Web crawler	F. Depth-first search
___ Google maps	G. Breadth-first search
___ PERT/CPM (Program Evaluation and Review Technique / Critical Path Method).	H. Dijkstra's algorithm
	I. Topological sort
	J. Bellman-Ford
	K. Enumerate permutations

8. Dijkstra's algorithm



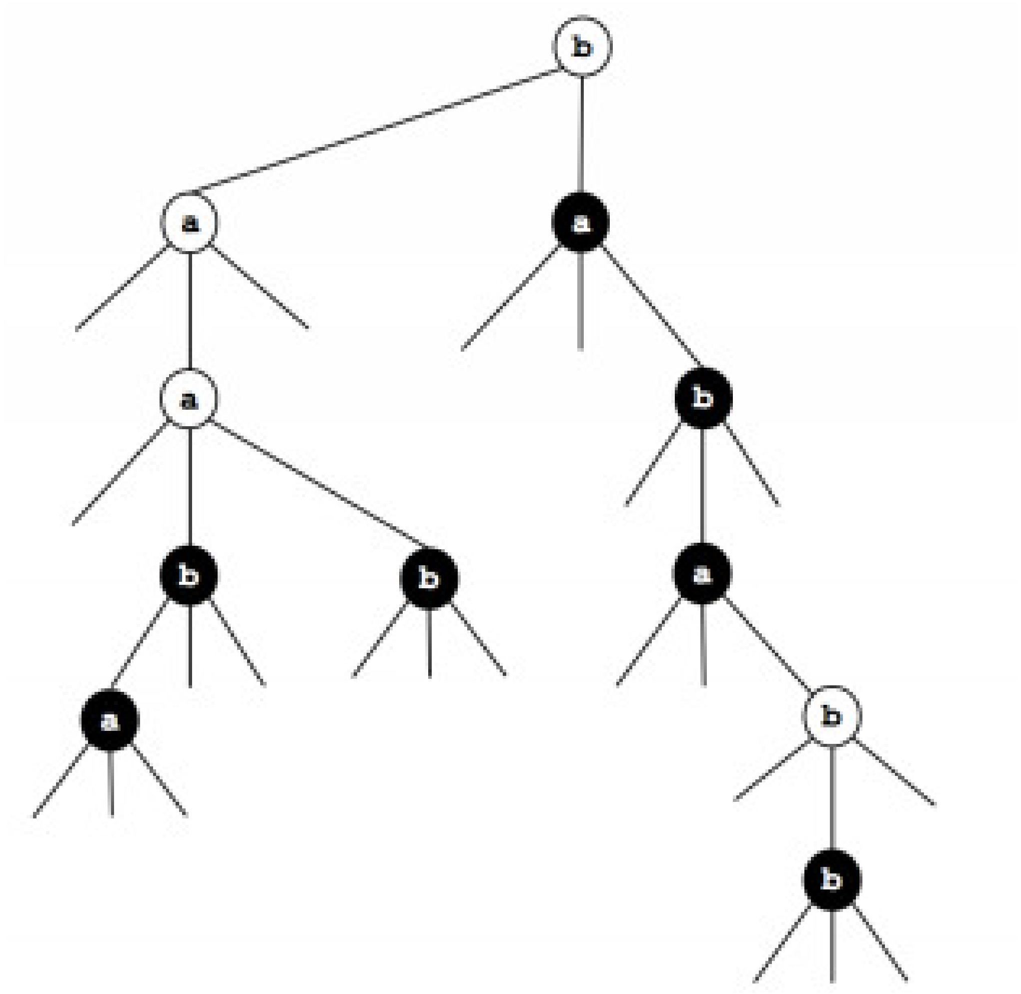
v	distTo[]	edgeTo[]
0		
1		
2		
3		
4		
5		
6		
7		

Challenge problems

- Answer each question as possible, impossible or unknown
  - Find the strong components in a digraph in linear time
  - Construct a binary heap in linear time
  - Find the maximum spanning tree in time proportional to E+V

Strings

9. TST



1. List the words in alphabetical order (black nodes denote the end of a word)

2. Insert aaca to TST

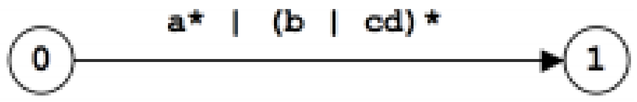
10. String Sorting

Put an X in each box if the string sorting algorithm (the standard version considered in class) has the corresponding property.

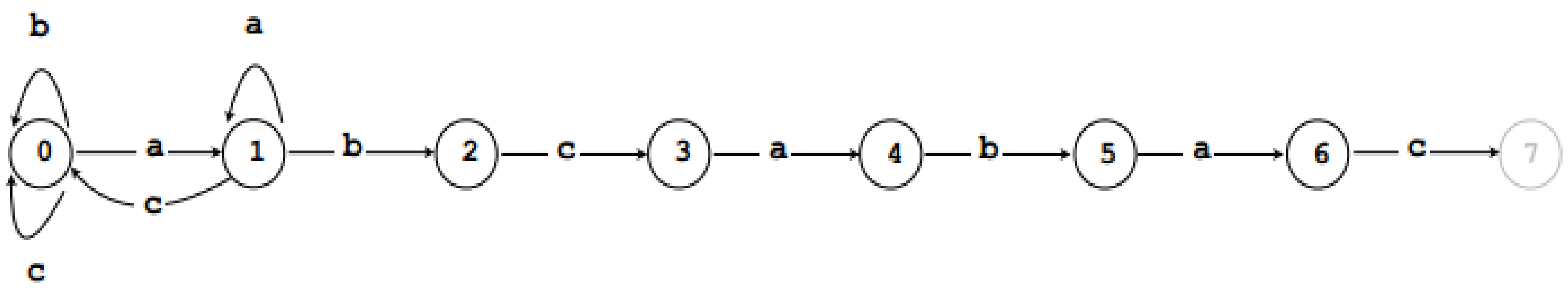
	mergesort	LSD radix sort	MSD radix sort	3-way radix quicksort
stable				
in-place				
sublinear time (in best case)				
fixed-length strings only				

12. Regular Expression to NFA

Convert the RE `a* | (b | c d)*` into an equivalent NFA using the algorithm described in lecture, showing the result after applying each transformation.



13. KMP Table



Compression

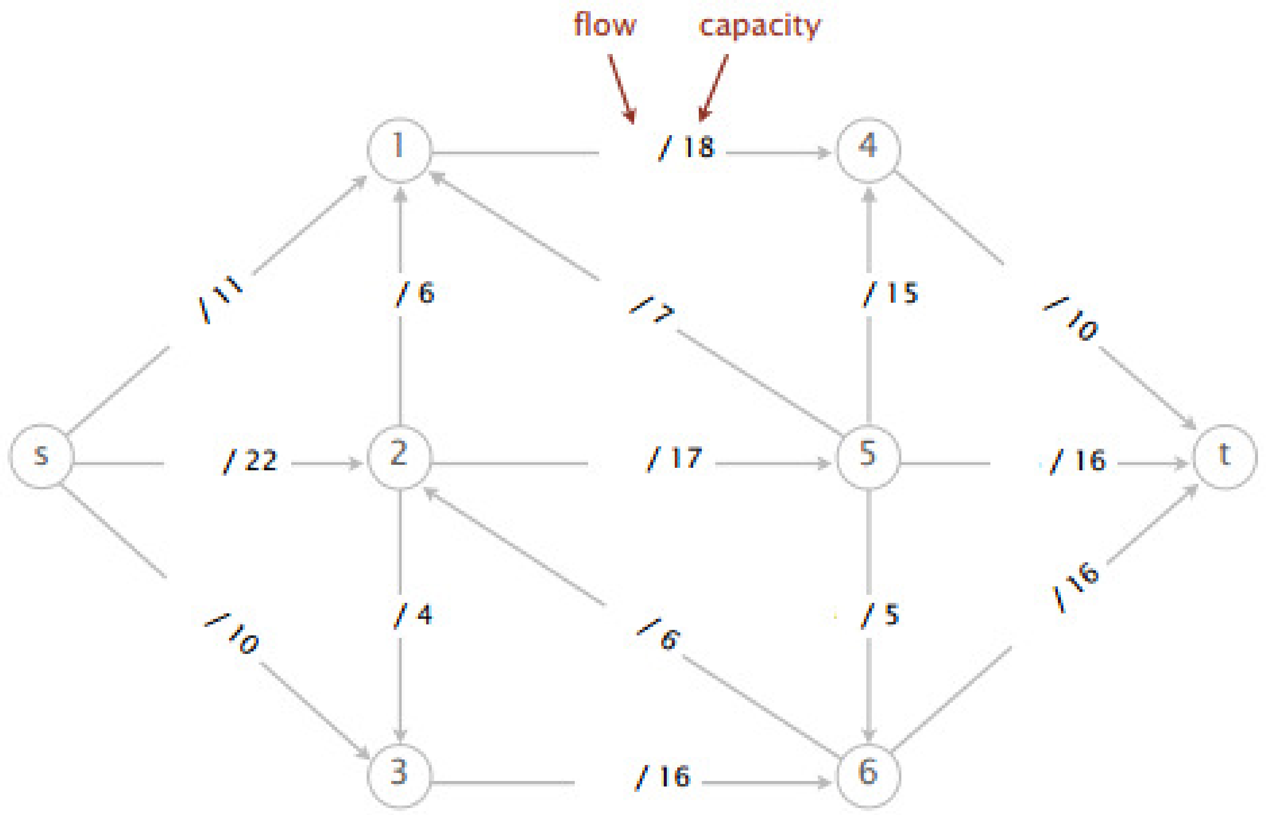
14. LZW compression

You receive the following message encoded using LZW compression.

```
 97
 98
128
129
131
132
130
```

Decode the message (a -97, b-98 … and next code starts - 128

15. MaxFlow-MinCut



17. Algorithm Design

In data compression, a set of binary code words is *prefix-free* if no code word is a prefix of another. For example, $\{01, 10, 0010, 1111\}$ is prefix free, but $\{01, 10, 0010, 10100\}$ is not because 10 is a prefix of 10100.

1. Design an efficient algorithm to determine if a set of binary code words is prefix-free

2. What is the order of growth of the worst-case running time of your algorithm as a function of N and W, where N is the number of binary code words and W is the total number of bits in the input?

3. What is the order of growth of the memory usage of your algorithm?

19. Burrows-Wheeler

What is the Burrows-Wheeler transform of

```
b   a   b   a   a   b   a   c
```

What is the Burrows-Wheeler inverse transform of

```
7
b   b   b   b   a   a   a   a   a
```

22. String Sorting

KISS	ABBA	ENYA	ABBA	ENYA	ACDC	SOAD	SADE	ABBA
ENYA	ACDC	INXS	ACDC	ABBA	ABBA	WHAM	CAKE	ACDC
INXS	AQUA	DIDO	AQUA	AQUA	AQUA	ABBA	CARS	AQUA
STYX	BECK	CARS	BECK	ACDC	BUSH	MOBY	JAYZ	BECK
SOAD	BLUR	ACDC	BLUR	SOAD	BLUR	BECK	ABBA	BLUR
ACDC	BUSH	FUEL	BUSH	CAKE	BECK	ACDC	ACDC	BUSH
KORN	CAKE	BUSH	CAKE	MUSE	CAKE	SADE	BECK	CAKE
FUEL	CARS	ABBA	CARS	HOLE	CARS	DIDO	WHAM	CARS
BUSH	DIDO	AQUA	DIDO	SADE	DIDO	FUEL	DIDO	DIDO
ABBA	ENYA	CAKE	ENYA	BUSH	ENYA	CAKE	KISS	ENYA
WHAM	FUEL	BLUR	FUEL	RUSH	FUEL	HOLE	BLUR	FUEL
CAKE	HOLE	JAYZ	HOLE	BECK	HOLE	TSOL	INXS	HOLE
BLUR	INXS	BECK	INXS	FUEL	INXS	KORN	ENYA	INXS
MUSE	JAYZ	HOLE	JAYZ	TSOL	JAYZ	CARS	SOAD	JAYZ
BECK	KISS	KORN	KISS	WHAM	KISS	MUSE	MOBY	KISS
MOBY	KORN	KISS	KORN	KORN	KORN	BUSH	HOLE	KORN
HOLE	MUSE	TSOL	TSOL	DIDO	MUSE	RUSH	KORN	MOBY
TSOL	MOBY	MOBY	MOBY	BLUR	MOBY	KISS	AQUA	MUSE
JAYZ	RUSH	MUSE	MUSE	KISS	RUSH	AQUA	TSOL	RUSH
AQUA	STYX	SADE	SADE	INXS	STYX	BLUR	STYX	SADE
SADE	SOAD	WHAM	WHAM	CARS	SOAD	INXS	FUEL	SOAD
CARS	SADE	SOAD	SOAD	STYX	SADE	ENYA	MUSE	STYX
DIDO	TSOL	RUSH	RUSH	MOBY	TSOL	STYX	BUSH	TSOL
RUSH	WHAM	STYX	STYX	JAYZ	WHAM	JAYZ	RUSH	WHAM
----	----	----	----	----	----	----	----	----
0								1

(0) Original input

(1) Sorted

(2) LSD radix sort

(3) MSD radix sort

(4) 3-way string quicksort (no shuffle)

KMP Table

Complete the DFA using partial information given

	0	1	2	3	4	5	6	7	8	9	10
A	0	0								10	11
B					5		2				4
s										A	

23. Reductions

Consider the following two problems:

- 3SUM. Given N integers $x_1, x_2, \ldots, x_N$, are there three distinct indices i, j, and k such that $x_i + x_j + x_k = 0$?

- 3SUMPLUS. Given N integers $x_1, x_2, \ldots, x_N$ and an integer b, are there three distinct indices i, j, and k such that $x_i + x_j + x_k = b$?

(a) Show that 3SUM linear-time reduces to 3SUMPLUS. To demonstrate your reduction, give the 3SUMPLUS instance that you would construct to solve the following 3SUM instance: $x_1, x_2, \ldots, x_N$.

(b) Show that 3SUMPLUS linear-time reduces to 3SUM. To demonstrate your reduction, give the 3SUM instance that you would construct to solve the following 3SUMPLUS instance: $b, x_1, x_2, \ldots, x_N$.