COS 226
Final Exam Review
Spring 2016

Ananda Gunawardena
guna@cs.Princeton.edu

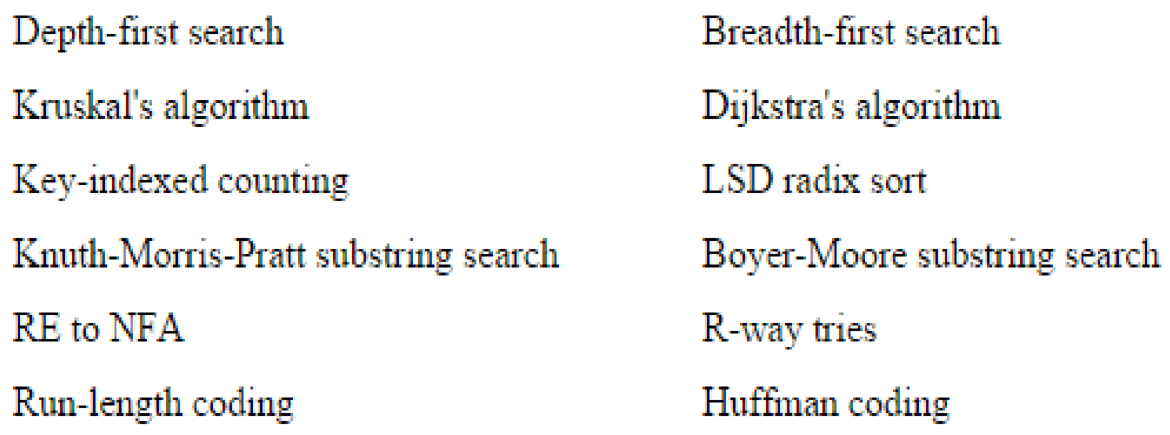
Logistics

- The final exam **time and location**
  - **Thursday May 19th from 9AM-12PM.**
  - Location
    - McCosh Hall 46: **Friday Precepts** P06, P07, P07A.
    - McCosh Hall 50: **Thursday Precepts** P01, P02, P02A, P03, P03A, P04, P05 and P99(flipped).
    - .
- **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).

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

Kruskal's algorithm

Key-indexed counting

Knuth-Morris-Pratt substring search

RE to NFA

Run-length coding

Breadth-first search

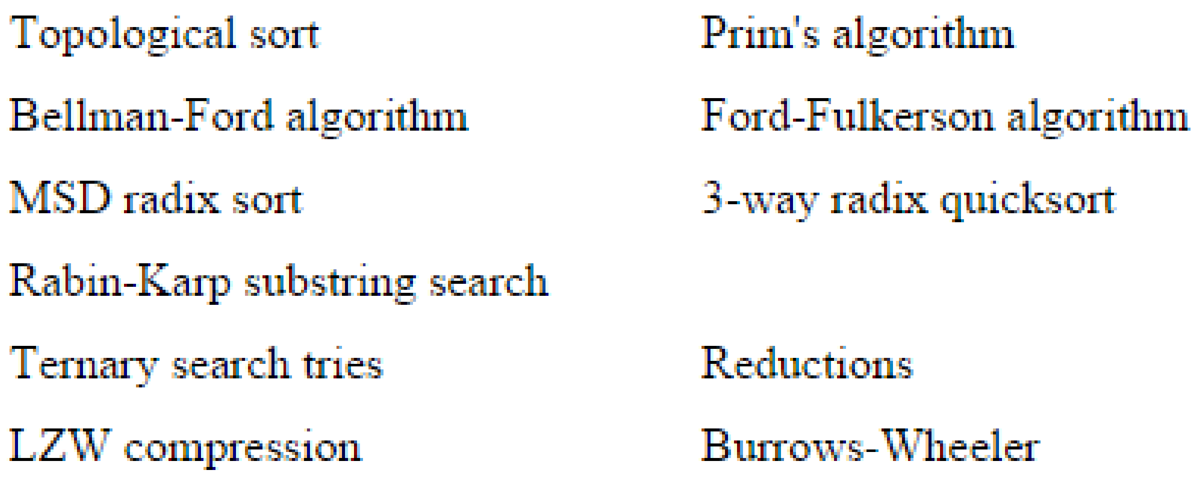
Dijkstra's algorithm

LSD radix sort

Boyer-Moore substring search

R-way tries

Huffman coding



Topological sort

Bellman-Ford algorithm

MSD radix sort

Rabin-Karp substring search

Ternary search tries

LZW compression

Prim's algorithm

Ford-Fulkerson algorithm

3-way radix quicksort

Reductions

Burrows-Wheeler

Algorithms

- focus on understanding fundamentals, not memorizing details (eg: code)
- *Write down as **many algorithms** as you can recall*
- ***For each algorithm***
  - understand how it works on typical, worst case, best case input
  - How is it different from other algorithms for the same class of problems?
  - When is it effective?

Algorithms	Application Area
KMP, Boyer-Moore, Rabin-Karp	String search
Insertion, selection, mergesort, quicksort, 3-way quicksort	General Sorting
LSD, MSD, 3-way MSD	String/Radix Sorting
Linear search, binary search	General Search
Kosaraju-Sharir	Strong components in a directed graph
Topological Sort	Vertex ordering
DFS, BFS	Graph search, path detection, cycle detection
Dijkstra’s, Bellman-Ford	Single-source shortest path
Ford-Fulkerson	MaxFlow-MinCut, Matching Algorithms
Huffman, run-length encoding, LZW, Burrows-Wheeler	Data compression
Prims (eager, lazy), Kruskal’s	Minimum Spanning Tree (MST)

Data Structures

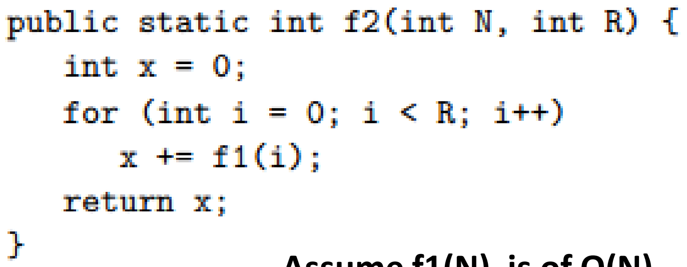
- *Write down as many data structures as you can*
- ***For each data structure***
  - How is it implemented?
  - Alternate implementations?
  - What type of problems are good for this data structure?
  - What are the memory requirements?

Data Structure	Properties
Arrays	Random access, contiguous memory, static
Resizable arrays	Random access, contiguous memory, dynamic with constant amortized cost per insert/delete
Linked Lists, Doubly Linked Lists	Sequential access, flexible memory allocation/deallocation
Priority Queues	Binary heap implementation, delMax, insert in worst case O(log N) time
Binary Search Trees	At most two children per node, worst case linear access, ordered
Red-Black Trees	Balanced BST, guaranteed log N operations
Kd-Trees	Efficient organization of multi-dimensional data
Directed Graphs	Set of vertices, set of directed edges
Undirected Graphs	Set of vertices, set of undirected edges
Tries	Supports efficient prefix lookup, store strings with common prefixes efficiently
Union-Find	Support two operations, union and find, implementations of quick union, quick find, weighted UF
Symbol tables	A key-value mapping, multiple implementations with red-black trees(ordered ST), arrays
Hash table	Constant time insert/search with uniform hashing, linear time worst case

Past Exams Problem Clusters

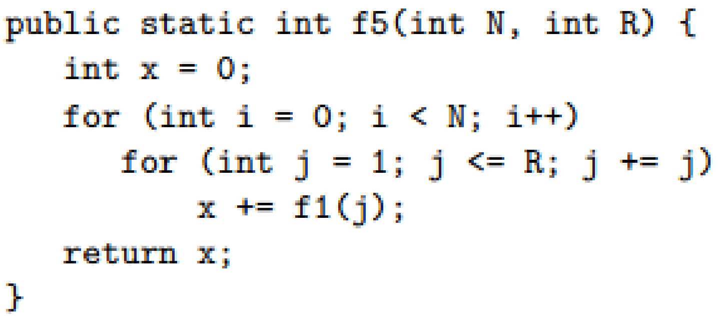
Identify the sort (Sorting invariants)	**Maxflow-mincut** (application of the algorithm)
Tilde Notation (counting comparisons, exchanges) **Order of growth Analysis** (recurrences)	**Design Problems** (data structure, algorithm, performance requirements)
Memory Calculation (size of data)	**Graph Algorithms Trace** (DFS, BFS, Kruskals, Prim's, Dijkstra's, SCA)
String Sort/Search (LSD, MSD, 3-way quicksort, Key-indexed counting, KMP, DFA construction, Boyer-Moore, Rabin-Karp)	**RegEx/NFA** (construction, tracing)
Tries and TST (construction, operations, memory)	**Compression** (Huffman, LZW compress/expand, Burrows-Wheeler)
Reductions (3sum→3sumvariant, longest path→ longest cycle)	**Miscellaneous** (Matching, True/False, Possible/impossible)

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;
}
```

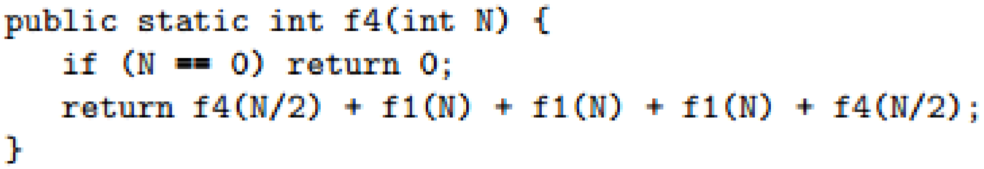
Assume f1(N) is of O(N)



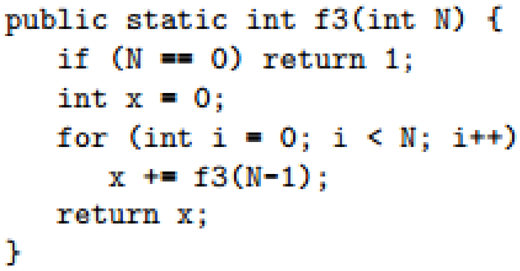
```
public static int f5(int N, int R) {
   int x = 0;
   for (int i = 0; i < N; i++)
      for (int j = 1; j <= R; j += j)
         x += f1(j);
   return x;
}
```

Analysis of Algorithms ctd..

Assume f1 is of O(N)

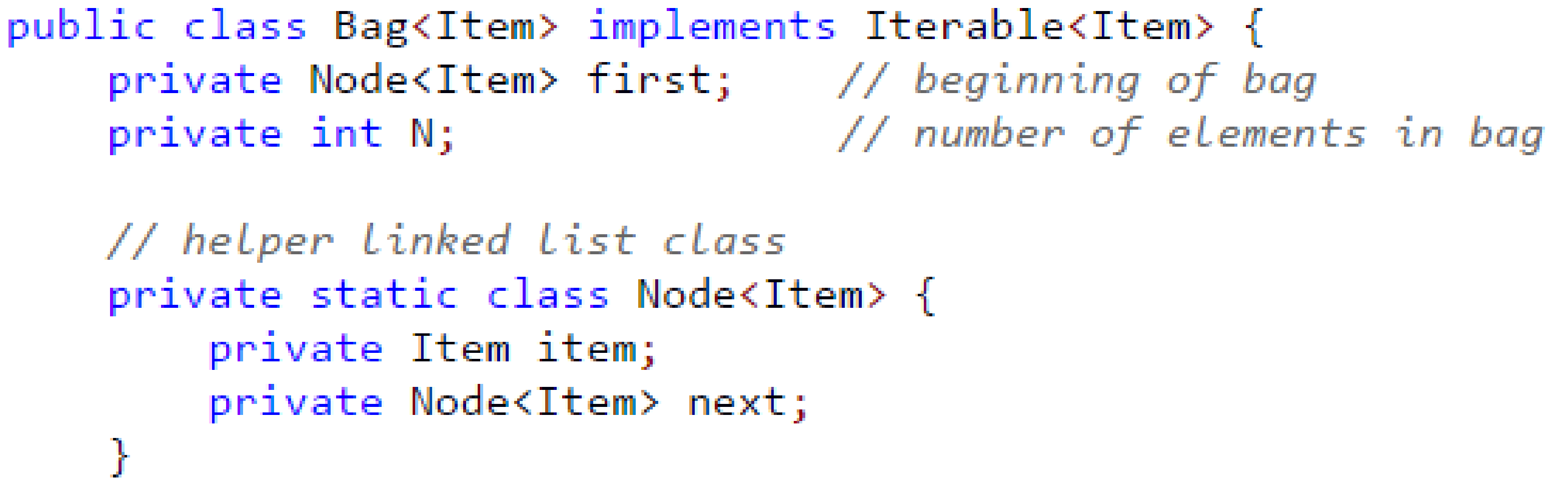


```
public static int f4(int N) {
   if (N == 0) return 0;
   return f4(N/2) + f1(N) + f1(N) + f1(N) + f4(N/2);
}
```



```
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;
}
```

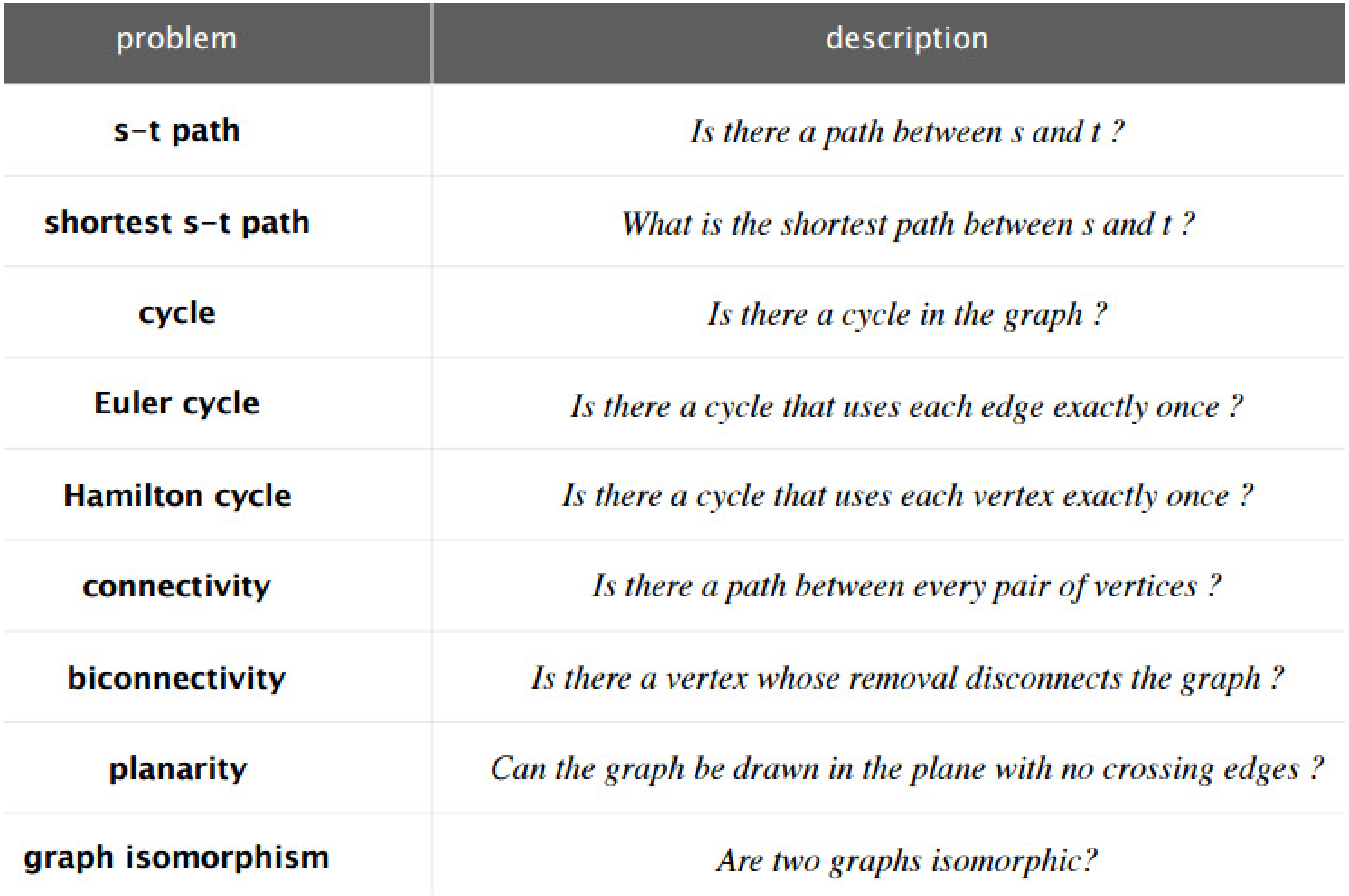
Counting Memory



```
public class Bag<Item> implements Iterable<Item> {
    private Node<Item> first;    // beginning of bag
    private int N;               // number of elements in bag

    // helper linked list class
    private static class Node<Item> {
        private Item item;
        private Node<Item> next;
    }
```

Graphs



problem	description
s-t path	*Is there a path between s and t ?*
shortest s-t path	*What is the shortest path between s and t ?*
cycle	*Is there a cycle in the graph ?*
Euler cycle	*Is there a cycle that uses each edge exactly once ?*
Hamilton cycle	*Is there a cycle that uses each vertex exactly once ?*
connectivity	*Is there a path between every pair of vertices ?*
biconnectivity	*Is there a vertex whose removal disconnects the graph ?*
planarity	*Can the graph be drawn in the plane with no crossing edges ?*
graph isomorphism	*Are two graphs isomorphic?*

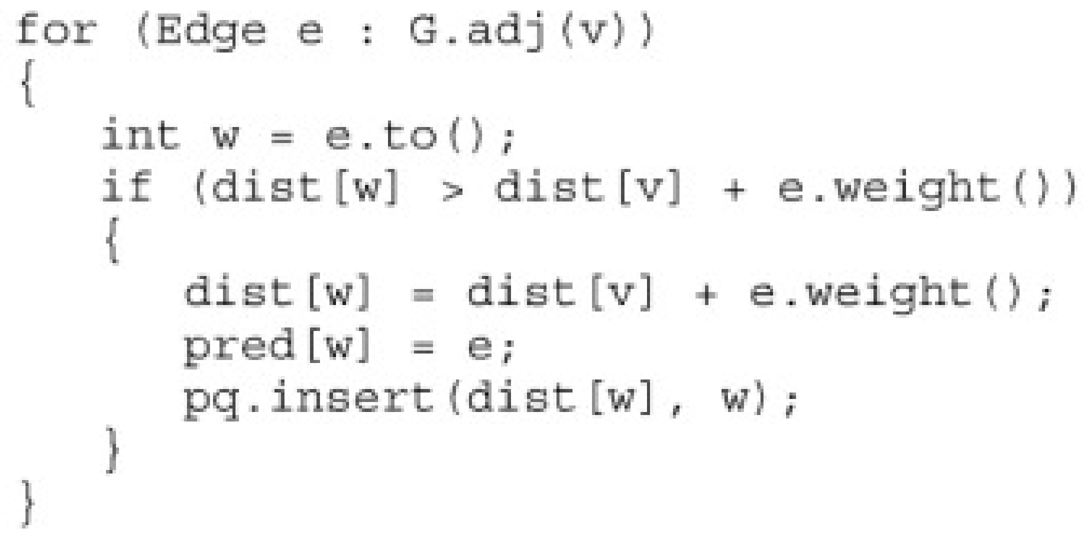
Graph Order Traversals

- **Preorder:** order in which dfs() is called.
- **Postorder:** order in which dfs() returns.
- **Reverse postorder:** reverse order in which dfs() returns.

DFS vs BFS

- DFS enables
  - Reachability.
  - Path finding.
  - Topological sort.
  - Directed cycle detection.
- BSF enables
  - Single source shortest path

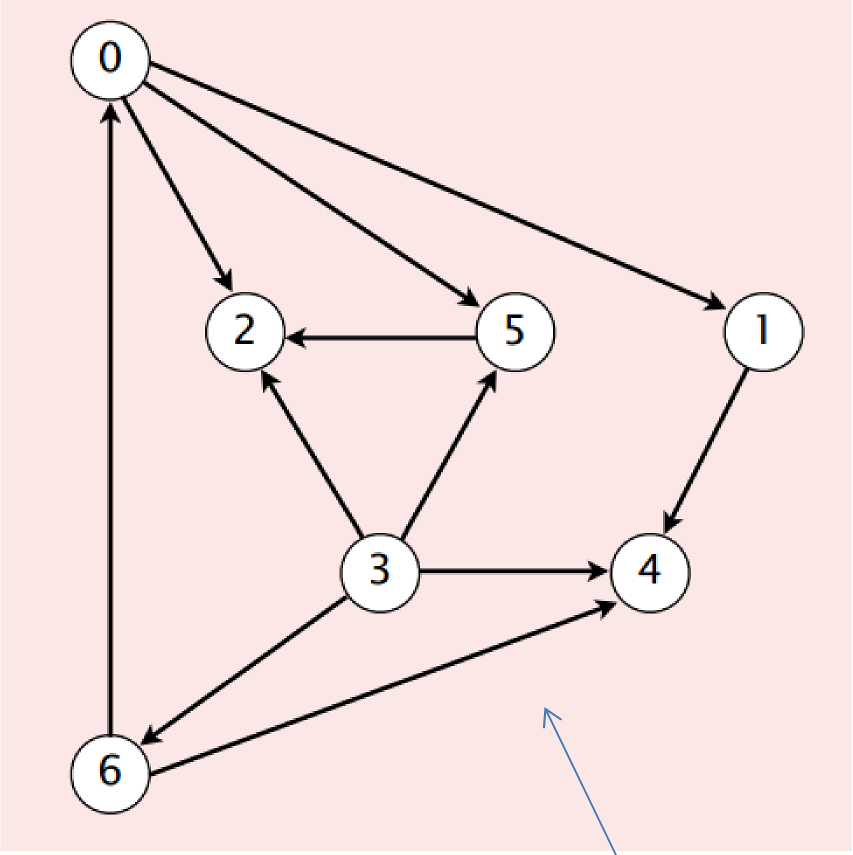
Mystery Code



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

Finding SCC's



v and w are **connected** if there is a path between v and w

v and w are **strongly connected** if there is both a directed path from v to w and a directed path from w to v

Find the Strongly Connected Components of this Graph

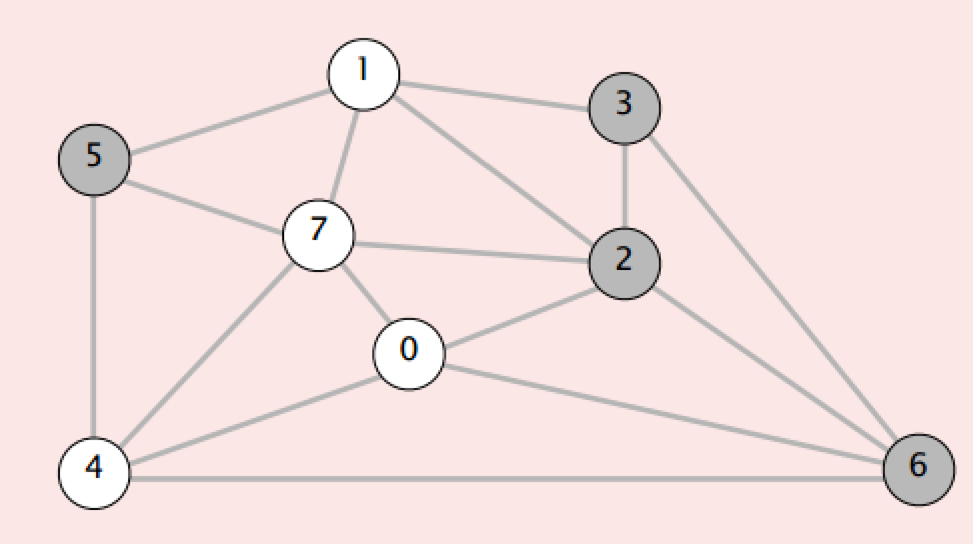
Hashing

- When implementing a ST with hashing, what operations are not allowed in the ST?
- What is a collision in a hashtable?
- How can we minimize collisions?
- With uniform hashing assumption, can we assure that hashtable operations always perform at O(1)

Hashing

- True/False
  - A linear probing hash table always finds a place
  - A separate chaining hash table always finds a place
  - The load factor of a hash table is always <= 1
  - A linear probing hash table must be rehashed if load factor is over 0.7
  - A rehashed entry will be at the same location as the original

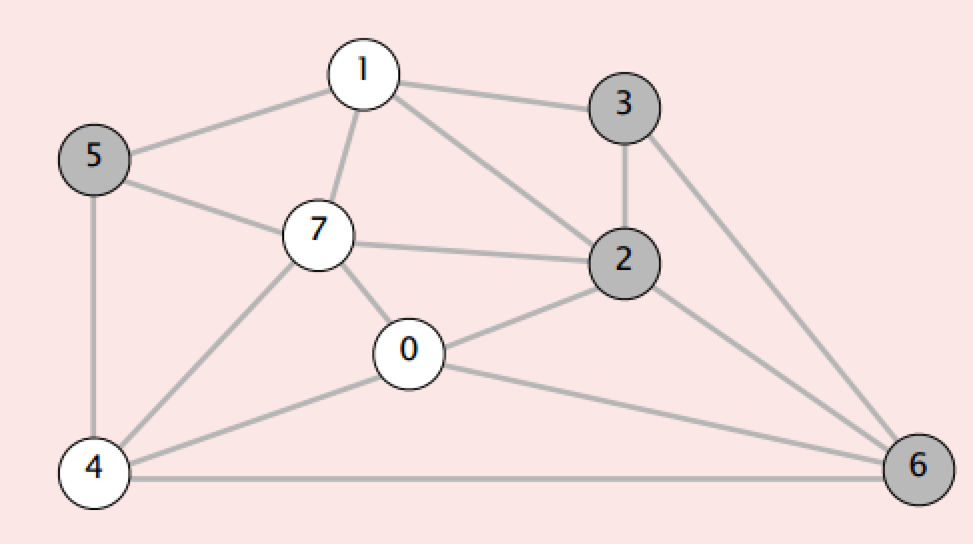
MST



0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

Which is the min weight edge crossing the cut
{ 2, 3, 5, 6 } ?

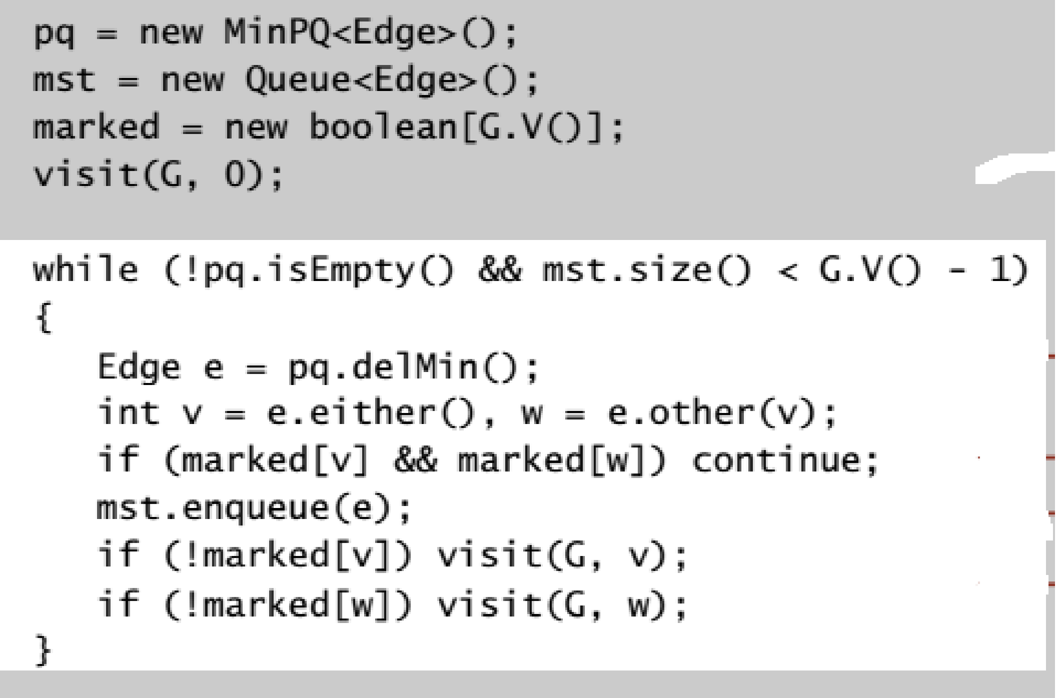
Kruskal’s



operation	frequency	time per op
build pq		
delete-min		
union		
connected		

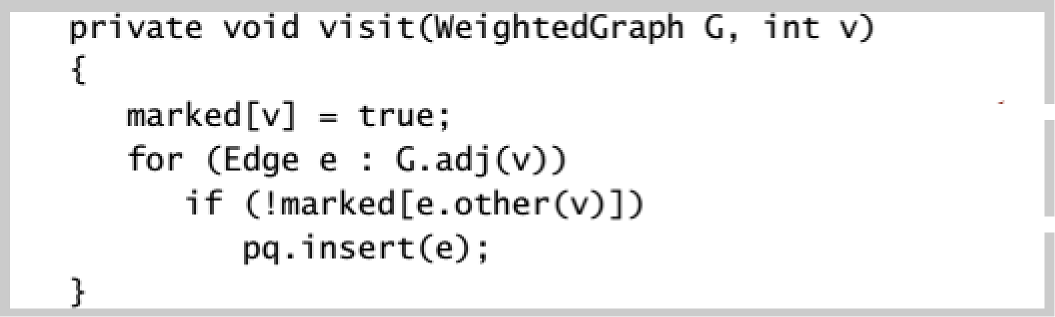
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

Prim’s (Lazy)



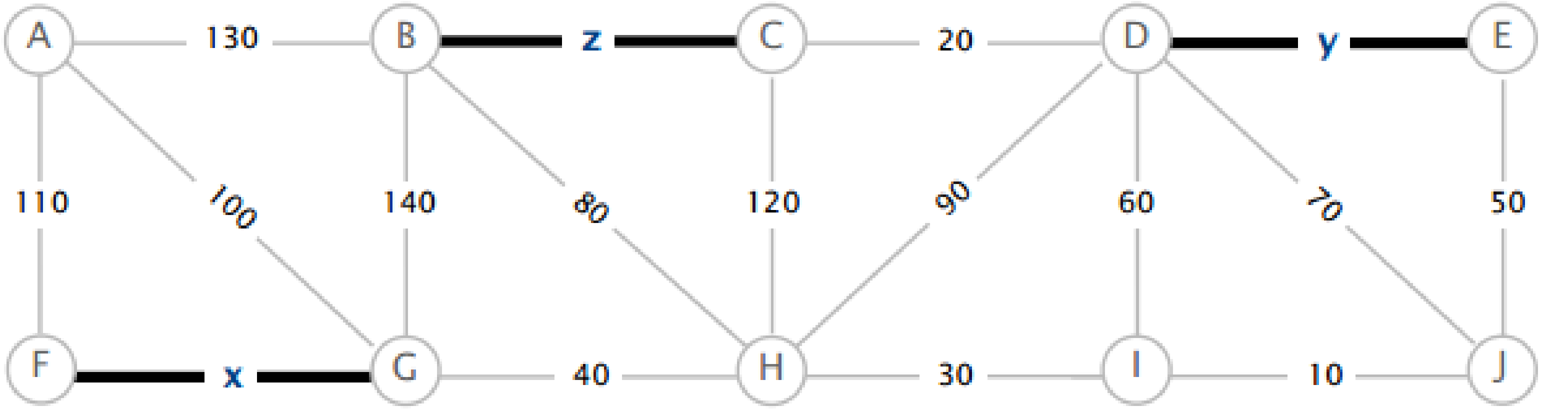
```
pq = new MinPQ<Edge>();
mst = new Queue<Edge>();
marked = new boolean[G.V()];
visit(G, 0);

while (!pq.isEmpty() && mst.size() < G.V() - 1)
{
   Edge e = pq.delMin();
   int v = e.either(), w = e.other(v);
   if (marked[v] && marked[w]) continue;
   mst.enqueue(e);
   if (!marked[v]) visit(G, v);
   if (!marked[w]) visit(G, w);
}
```



```
  private void visit(WeightedGraph G, int v)
  {
     marked[v] = true;
     for (Edge e : G.adj(v))
        if (!marked[e.other(v)])
           pq.insert(e);
  }
```

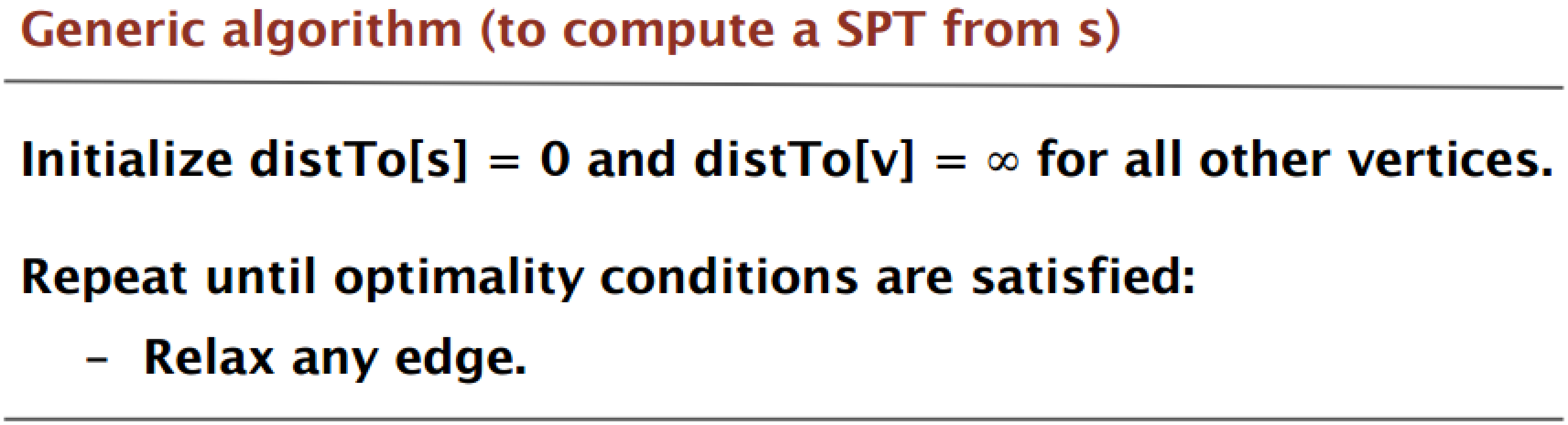
MST



If edges X, Y and Z are in the MST

1. Find the other edges that are in MST
2. find upper bounds for edge costs of X, Y and Z?

Shortest Paths



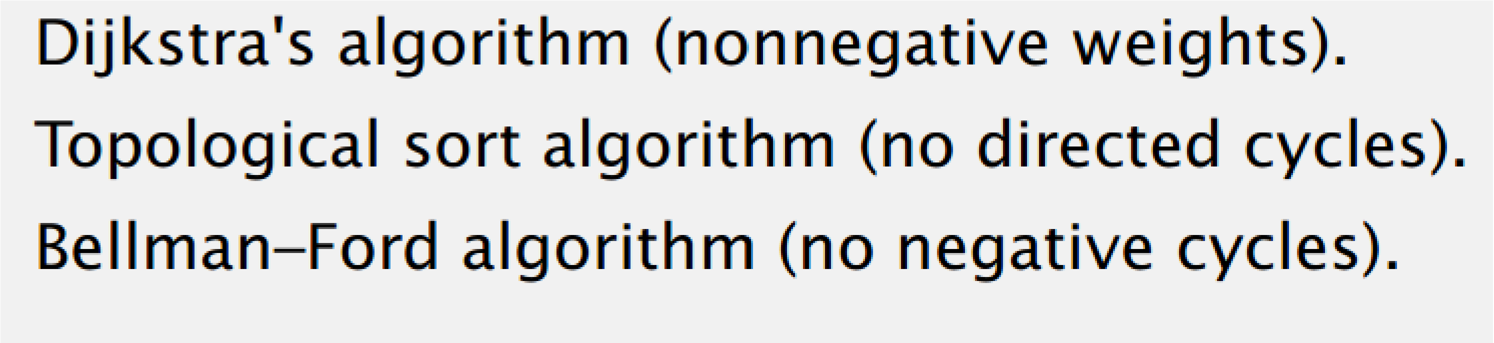
Generic algorithm (to compute a SPT from s)

Initialize distTo[s] = 0 and distTo[v] = ∞ for all other vertices.

Repeat until optimality conditions are satisfied:

- **Relax any edge.**

Specific Algorithms

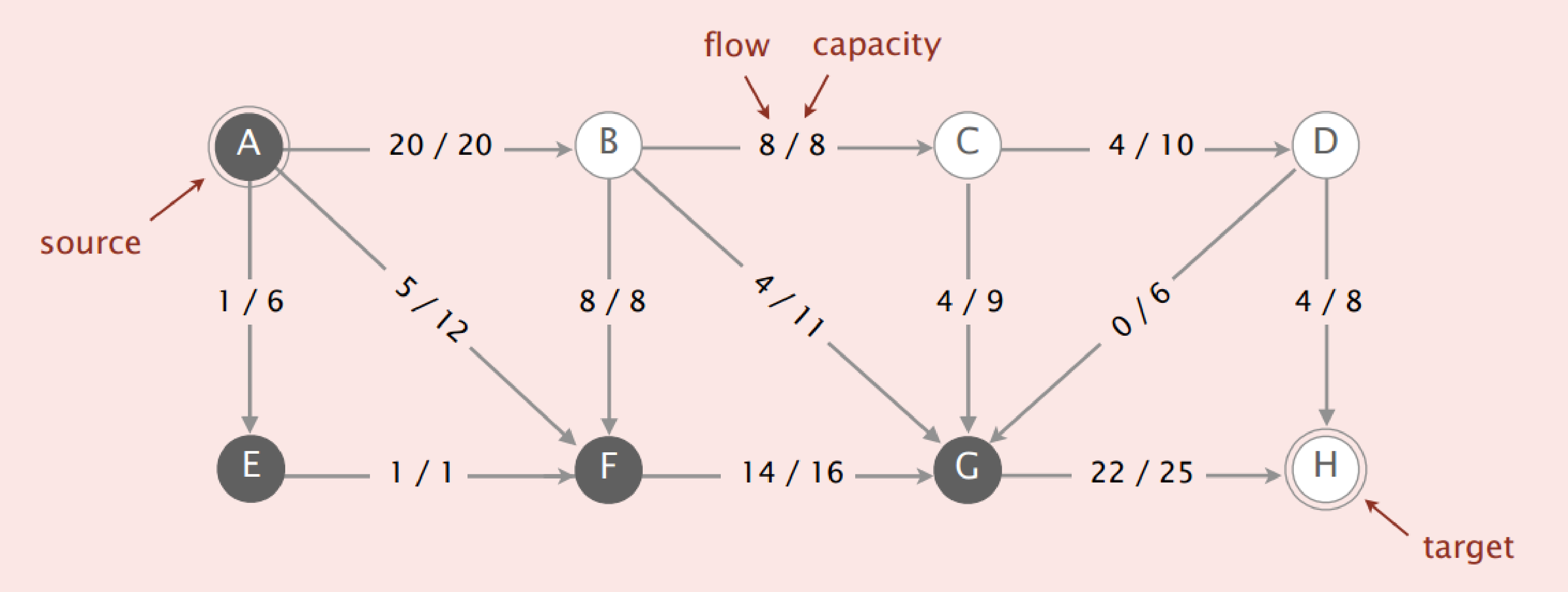


Dijkstra's algorithm (nonnegative weights).

Topological sort algorithm (no directed cycles).

Bellman–Ford algorithm (no negative cycles).

Max Flow



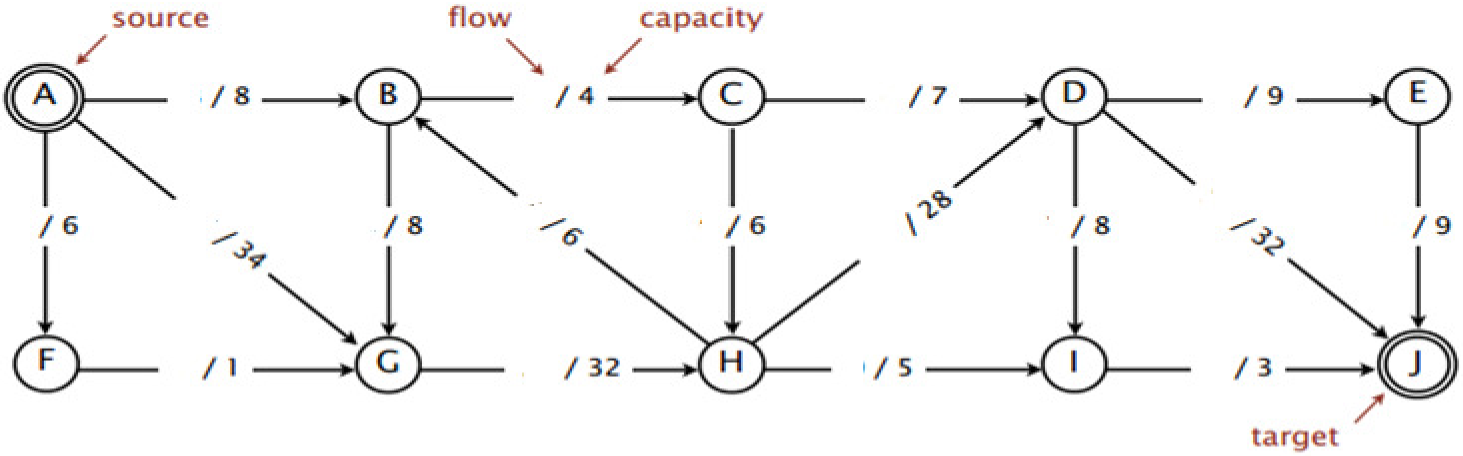
Fundamental questions.

- How to compute a mincut? Easy. ✔
- How to find an augmenting path? BFS works well.
- If FF terminates, does it always compute a maxflow? Yes. ✔
- Does FF always terminate? If so, after how many augmentations?

yes, provided edge capacities are integers
(or augmenting paths are chosen carefully)

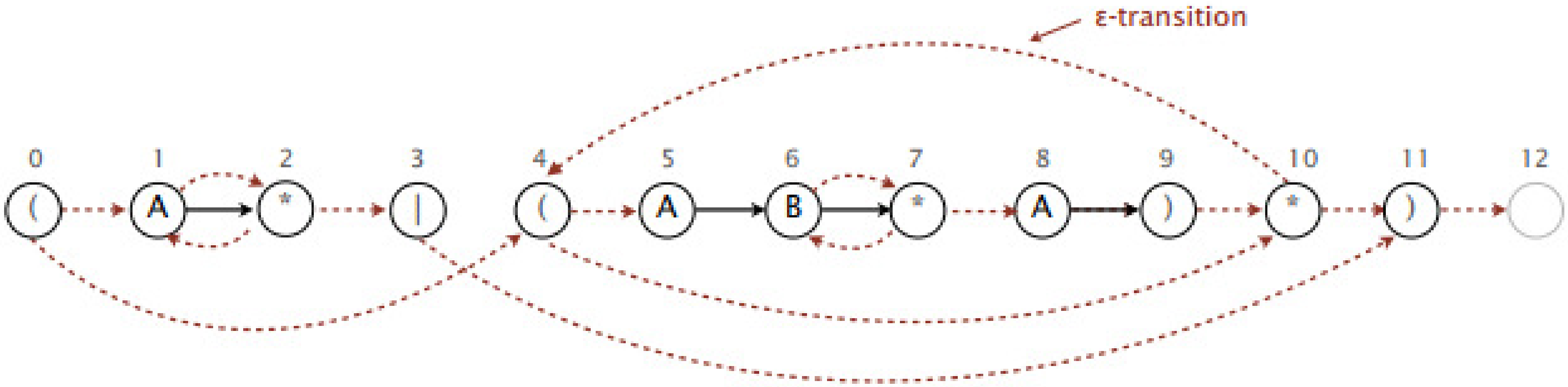
requires clever analysis

Ford-Fulkerson



1. What is the possible max flow of the network?
2. Mark an augmenting path and increase flow
3. Find all augmenting paths and increase flow
4. What is the actual max flow?
5. What is a min-cut?
6. Min-cut can only be calculated when a certain condition is true. What is it? How do we find out?

Regular Expressions and NFA



1. What is the regular expression?

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?

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
- Under what circumstances would you use this algorithm?
- If 8-bit words are used to store counts, what is the length of the maximum run that can be stored?
- What can we do if the length of the run cannot be accommodated by n-bit word?
- What is the best case input for run-length encoding (8-bit code words)

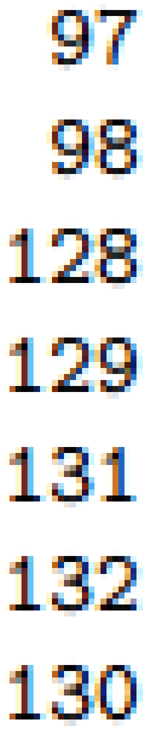
Compression Algorithms
Huffman encoding

- Describe the algorithm
- What is the preprocessing step of the algorithm?
- What data structure(s) is/are used in the preprocessing step?
- If you compress a file with all characters the same (eg: 10000 A's) what is the compression ratio?
- Describe a situation where no compression is obtained

Compression Algorithms
LZW encoding

- Describe the algorithm
- What data structure is used to store code words?
- Is it possible to run out of code words to store new words? How?
- Should we send the code words with the compressed file?
- How can we decompress a file?
- What is the tricky case and how do we overcome that?

LZW



97

98

128

129

131

132

130

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

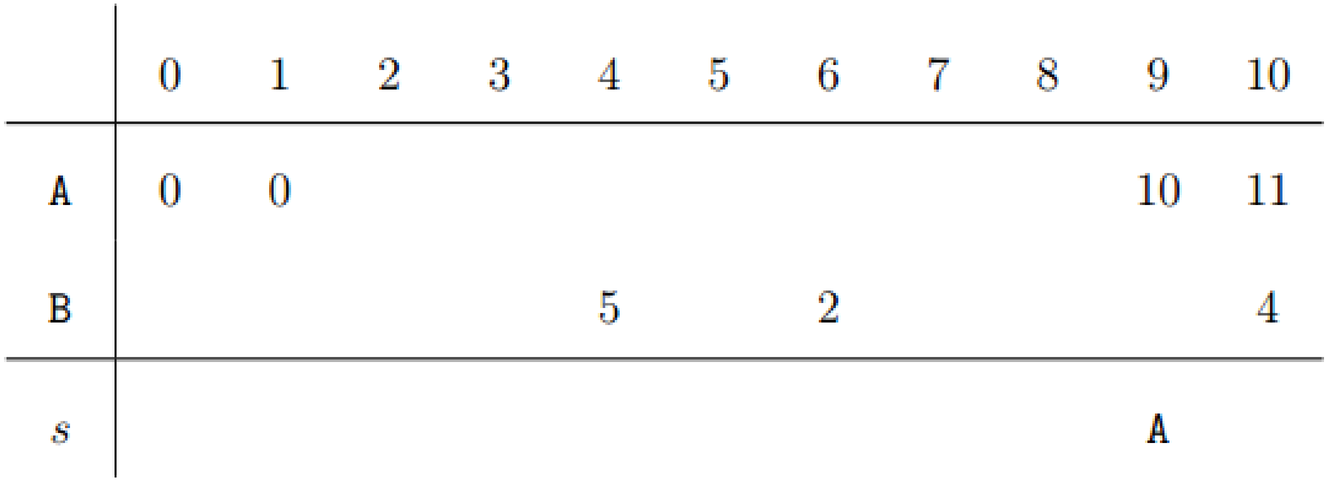
KMP Algorithm

- Briefly describe the algorithm
- What is the order of growth of building the DFA? Typical algorithm? Best algorithm?
- What is the order of growth of the algorithm for searching for a pattern of length m in a text of length n?
- Can KMP be adjusted to find all occurrences of a pattern in a text? What is the order of growth?
- Is KMP the algorithm of choice for any substring search application?

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										A	

Complete the table and find the search string

Boyer Moore (BM)

- Briefly describe the algorithm
- What is the pre-processing step of Boyer-Moore?
- 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 BW be adjusted to find all occurrences of a pattern in a text? What is the order of growth?

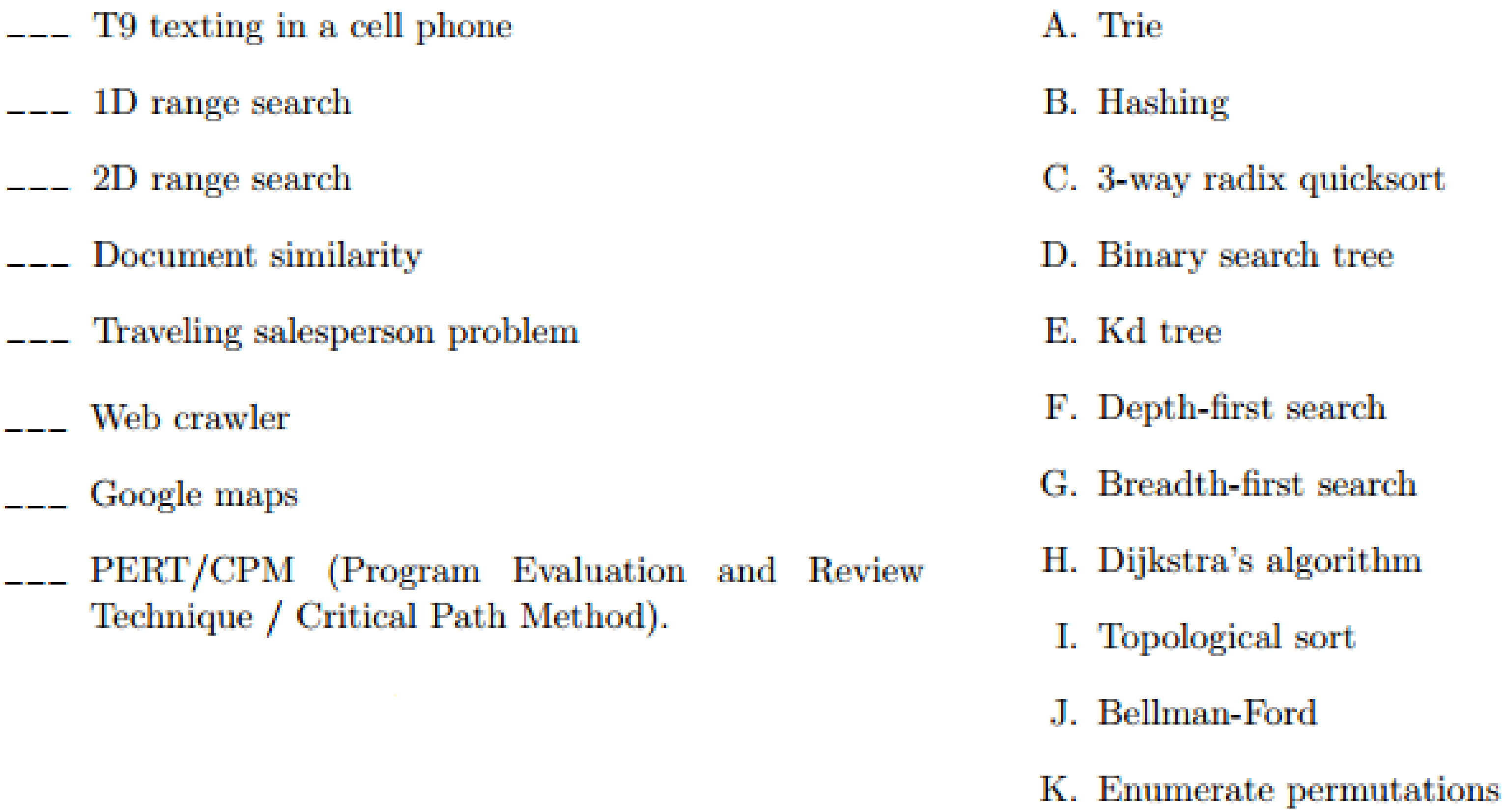
Rabin-Karp (RK)

- 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?

Challenge Questions

- **Answer each question with possible, impossible, unknown**
  - 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

Match Algorithms



___ T9 texting in a cell phone

___ 1D range search

___ 2D range search

___ Document similarity

___ Traveling salesperson problem

___ Web crawler

___ Google maps

___ PERT/CPM (Program Evaluation and Review Technique / Critical Path Method).

A. Trie

B. Hashing

C. 3-way radix quicksort

D. Binary search tree

E. Kd tree

F. Depth-first search

G. Breadth-first search

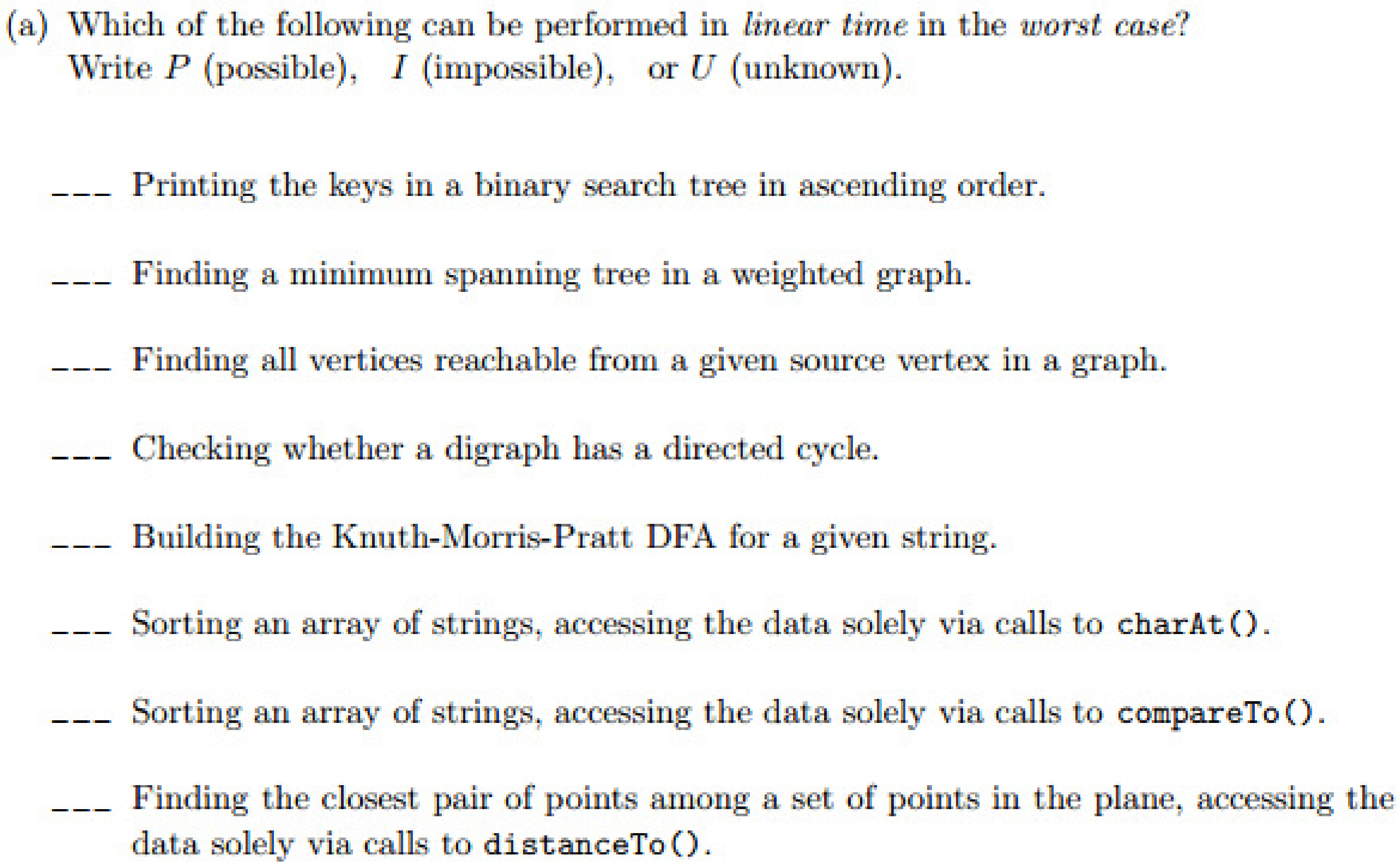
H. Dijkstra's algorithm

I. Topological sort

J. Bellman-Ford

K. Enumerate permutations

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).

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

___ Finding a minimum spanning tree in a weighted graph.

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

___ Checking whether a digraph has a directed cycle.

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

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

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

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

Wordnet

1. What data structures are used to store wordnet?

2. What data structures are used to store SCA?

3. Is there a reason that we used BFS not DFS?

4. What is the order of the best algorithm that can find the length of the common ancestor?

5. What is the order of the best algorithm that can find the common ancestor?

6. What is a rooted DAG and how do we determine that? Order or growth of your algorithm?

7. If the wordnet is NOT a rooted DAG, will answers to 3 and 4 will hold?

8. Given a list of n nouns, What is the order of growth of the outcast algorithm?

Seam-Carving

- What is the purpose of the seamcarving assignment?
- How does it relate to shortest path?
- How to find Vertical and Horizontal seams?
- Why do a defensive copy of Picture?
- What is the order of growth for the two methods, removeHorizontalSeam and removeVerticalSeam?

Burrows-Wheeler

- What libraries were used to read and write input/output to the program?
- What method in the output library that was required to print the output correctly?
- What data structures were used to implement BW, CSA, MoveToFront?
- What is the order of growth to form circular suffixes of a given string?
- What is wrong with using LSD.sort() in the program?
- Could we have used quicksort to sort suffixes? How? If so how can you avoid quadratic performance?

Burrows-Wheeler ctd…

- What are the 3-steps to burrows-wheeler transform?
- How would you sort strings w/o forming them explicitly?
- Is it necessary to do move to front? If not, why did we do it?
- How did we do the inverse transform?

Good Luck with All Finals