CS483 Analysis of Algorithms
Lecture 04 – Paths in Graph *

Jyh-Ming Lien

February 20, 2008

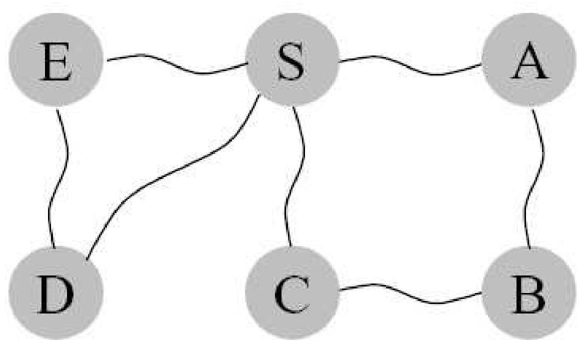
*this lecture note is based on *Algorithms* by S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani and *Introduction to the Design and Analysis of Algorithms* by Anany Levitin.

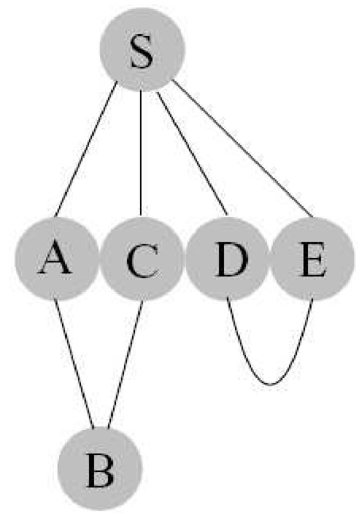
Distance of Graphs

- Distance between two nodes of a graph is the length of the shortest between them
- DFS is pretty bad for finding shortest paths
  - Example:

Single-source shortest-paths problem

- Given a graph $G = \{V, E\}$ and a start $s \in V$, find shortest paths from S to all the other vertices in G.
- Example:
- Another Intuition: Imagine the graph is made of tiny balls and strings:
  - Pick up the ball representing s high enough
  - The vertical distances from s to other balls are the shortest distances





Unweighted Graphs

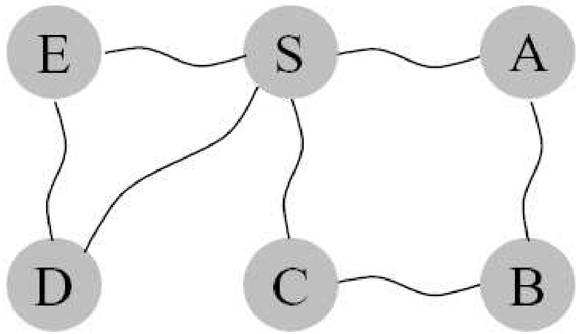
Unweighted Graphs

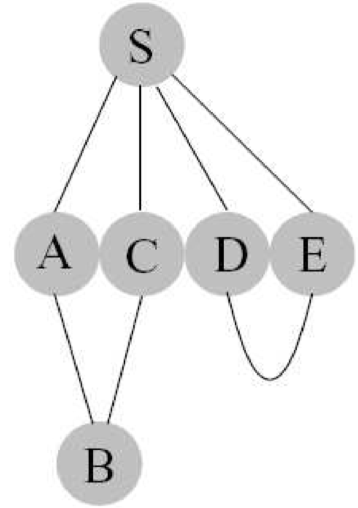
- For unweighted Graphs, breadth-first search (BFS) can find shortest path
- BFS mimics eactly the same idea of ball-string intuition mentioned above

Algorithm 0.1: $\mathrm{BFS}(G, v)$

BFS

- Examples:





- Related problem: Given a tree, how to print all tree nodes in the pre-order, post-order, level-order?

Weighted Graphs

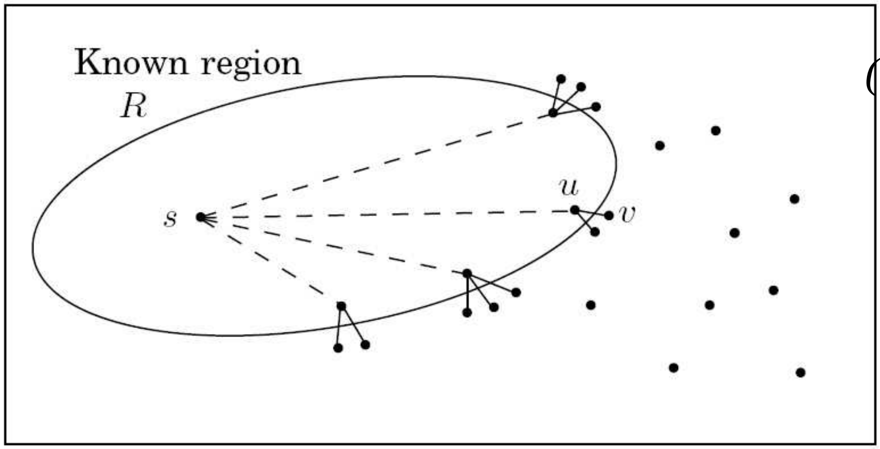
Will BFS still work?

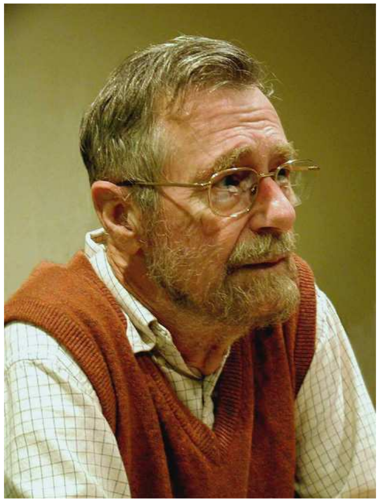
- No, but we can easily modify the graph to make BFS work! How?

- What data structure should we use to mimic this?

Dijkstra's algorithm

- Edsger Dijkstra (1930-2002): one of the most influential computer scientists
- Dijkstra's algorithm works by extending the current *shortest-paths tree* to the next closest vertex (to the source)
- Example:





(http://www.cs.utexas.edu/users/EWD/)

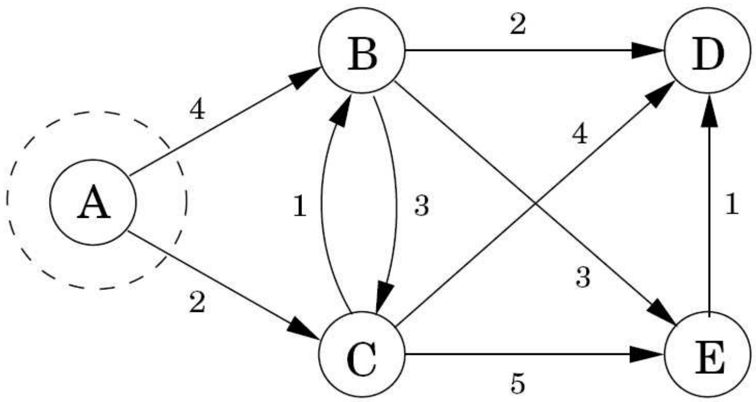
- Algorithm

Algorithm 0.2: DIJKSTRA$(G = \{V, E\}, s)$

- What data structure is needed to perform this algorithm?

Dijkstra's algorithm

- Example:



Notes on Dijkstra's algorithm

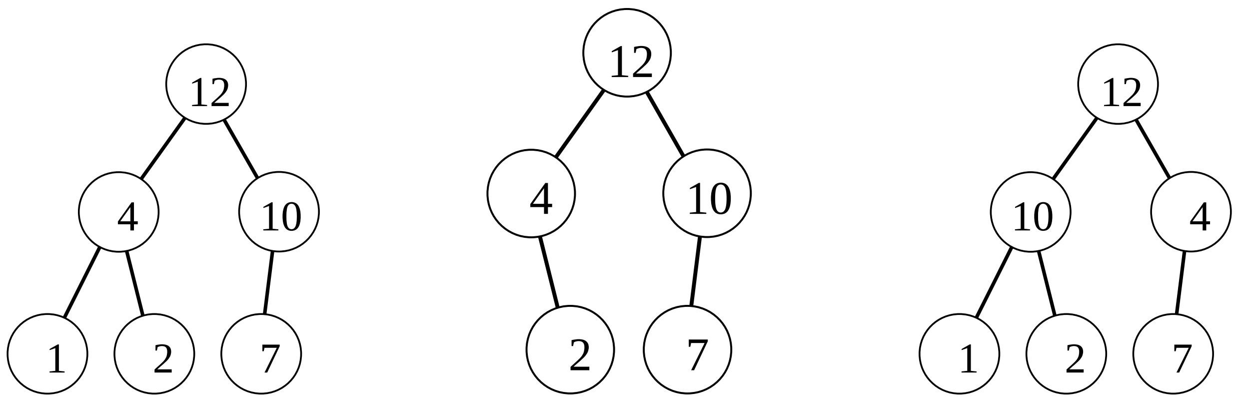
-
-
- Efficiency:
  -
  -

Priority Queue

- Consider problems that require you to:
  - schedule tasks (e.g., CPU)
  - match n men to n women (eHarmony.com)
  - route mails (Internet package routing)
- All these problems need to deal with dynamic data/information and contain information about priority/ordering/preference.
- A **priority queue** is needed in these problems to perform the following operations:
  - Find the element with the highest priority
  - Delete the element with the highest priority
  - Insert element
- Options for building a priority queue
  - a pointer points to the highest priority (what's the drawback?)
  - a sorted array (what's the drawback?)
  - a sorted list (what's the drawback?)
  - a balanced binary search tree (what's the drawback?)

Heap

- Heap is data structure good for building **priority queue**
  - Heap can be represented by a binary tree (and is usually stored in an array)
  - For each node n in a heap, n's key is always larger then the keys of n's kids (so, the largest value is in the root)
  - Only left most leaves are allowed (easier to expand, delete, and store)



- Keys in a heap is usually stored in an array
  - The kids of a node with index i have indices $2i$ and $2i + 1$
  - The parent of a node with index i have index $\lfloor \frac{i}{2} \rfloor$
  - Example: the heap above can be stored as $\{12, 4, 10, 1, 2, 7\}$

Heap: Insertion

- Assuming that we have a heap, and given a value with key k, insert the value to the heap.

Algorithm 0.3: HEAPINSERT(H, k)

- To build a heap, we can simply call HEAPINSERT iteratively.

Algorithm 0.4: HEAPBUILD($A[1 \cdots n]$)

Heap: Deletion

- Deleting the value with the highest priority can be done in a similar way

Algorithm 0.5: HEAPDELMAX(H)

- Example: Build a heap from this list: $\{2, 9, 7, 6, 5, 8\}$ and delete one element

Heap: Time Complexity

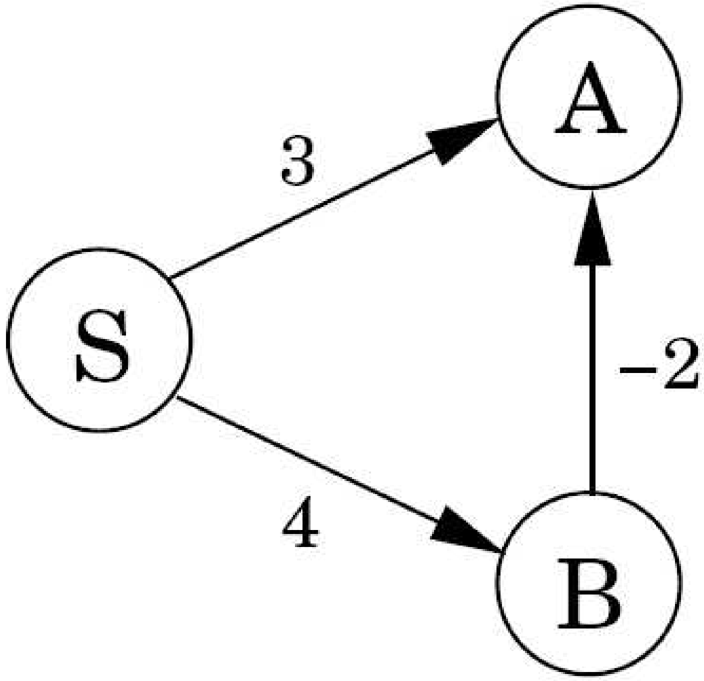
- What is the time complexity of
  - Find the max element
  - Insert an element
  - Delete an element

- heapsort: pop the largest element from the heap, i.e., call HEAPDELMAX $(n-1)$ times
  - time complexity

Graphs with Negative Weights

Will Dijkstra's algorithm still work?

- Example:



- Repetitively call update for all edges $|V| - 1$ times
  - Why do we pick $|V| - 1$?

Algorithm 0.6: BELLMAN-FORD(G, s)

- Does Bellman-Ford algorithm work for undirected graph?

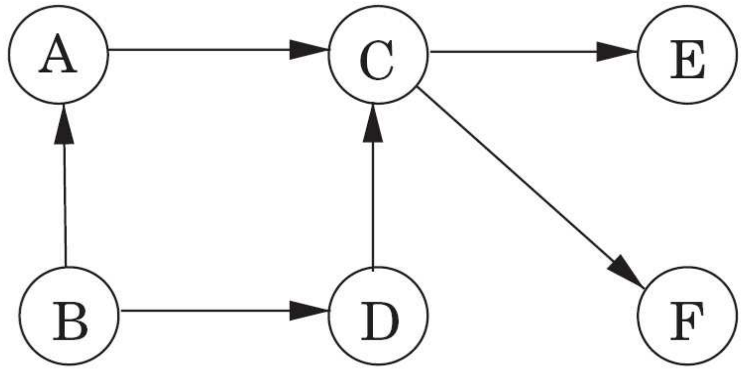
Negative Cycles

- Example:

- How do we discover or deal with negative cycles?

DAG

- Example:



- Algorithm

Algorithm 0.7: DAG-SHORTEST-PATH(G, s)

Conclusion

Summary

- Single-source shortest paths problem
  - unweighted graph
  - weighted graph with positive weights (Dijkstra's)
  - weighted graph with positive and negative weights (Bellman-Ford)
  - DAG
- Priority Queue
  - Heap
- Assignments:
  - 4.1
  - 4.2
  - 4.8
  - 4.12
  - 4.17