FINDING SHORTEST PATHS

Xinchen Yan

23th. Apr, 2012

SYSNOPSIS

- Purpose of Speech
- Brief Intro to Shortest Paths
- A Simple implementation: BFS
- SPFA: Optimized BFS
- Shortest Paths on Tree

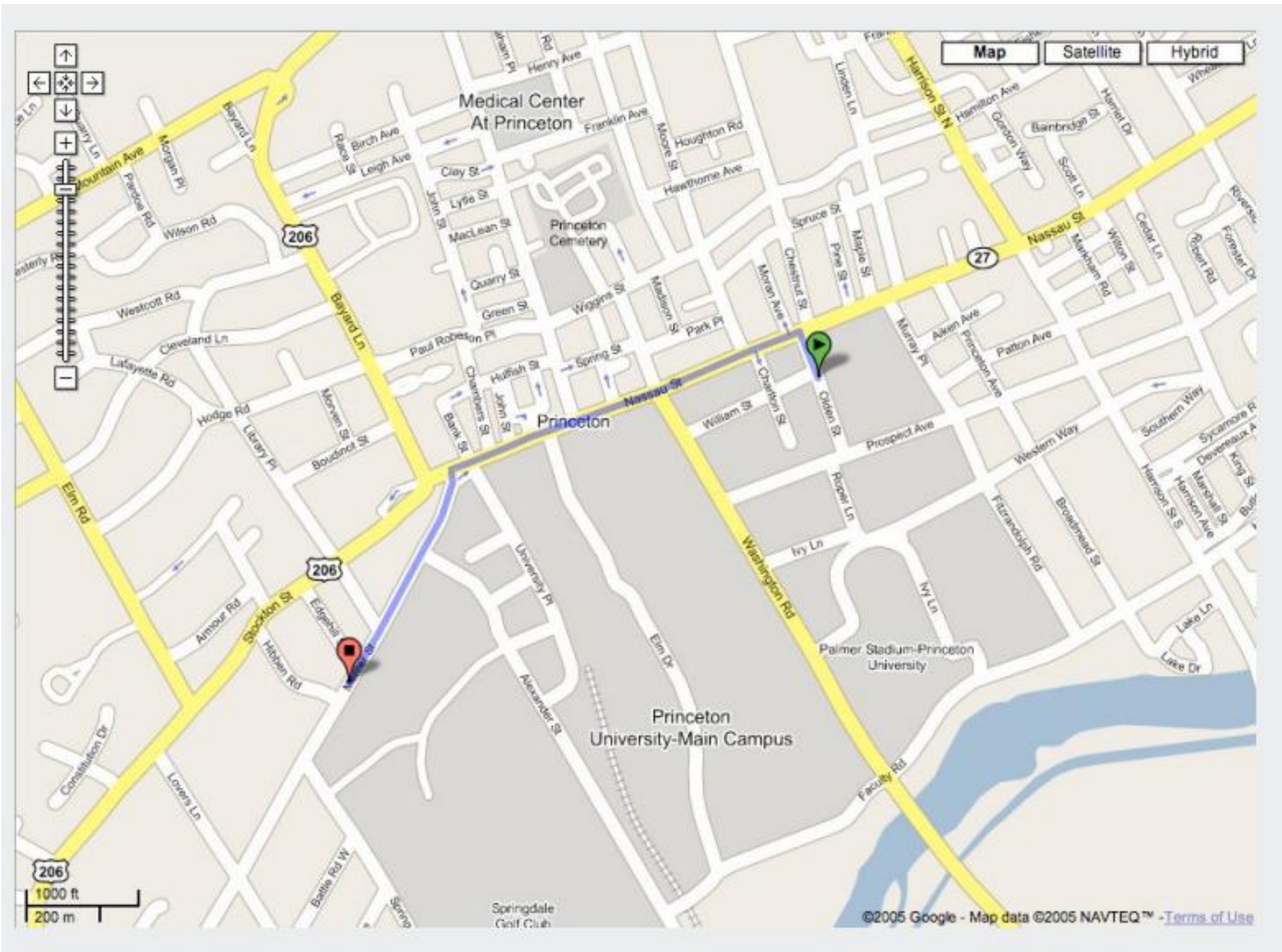
PURPOSE

- Fundamental skills for CS students
- A broadly useful problem solving model
- Frequent appearance in Computer-based Exams
- OJ 2101
- OJ 2106

APPLICATIONS

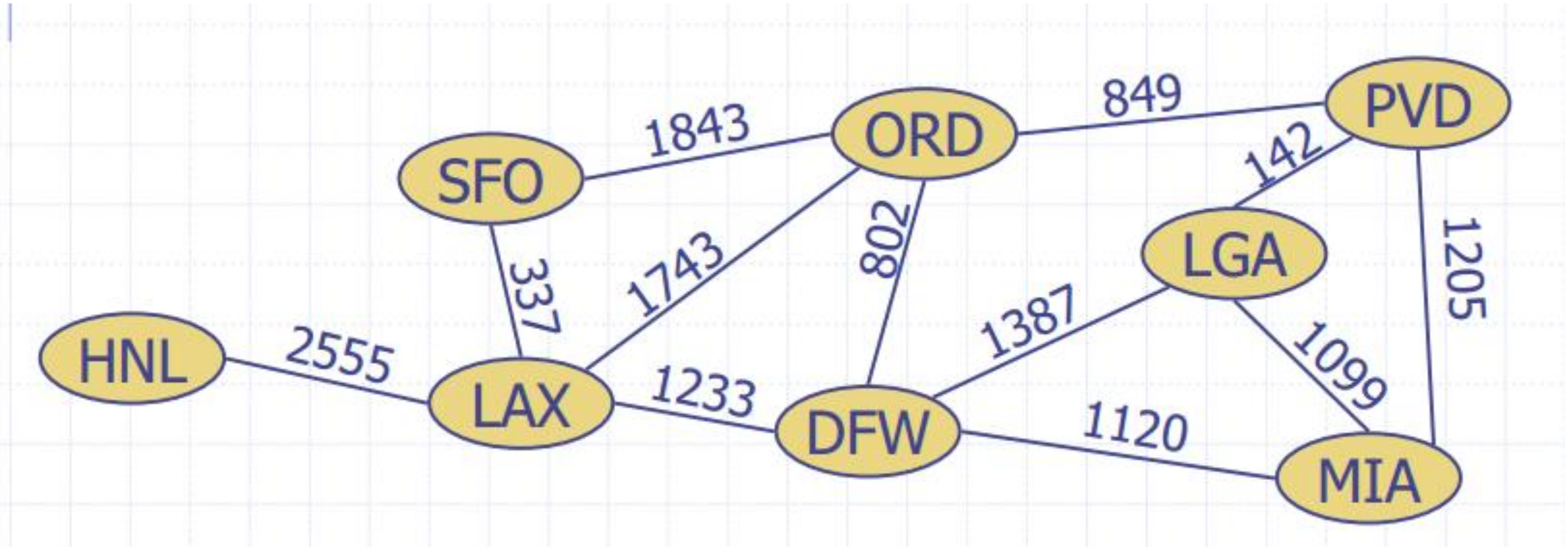
- Maps
- Robot navigation
- Texture mapping
- Typesetting in Tex
- Urban traffic planning
- Optimal pipeling of VLSI chip
- Telemarketer operator scheduling
- Network routing protocols

INTRO TO SHORTEST PATHS



INTRO TO SHORTEST PATHS

- Weighted Graphs
- Vertex
- Edge
- Distance

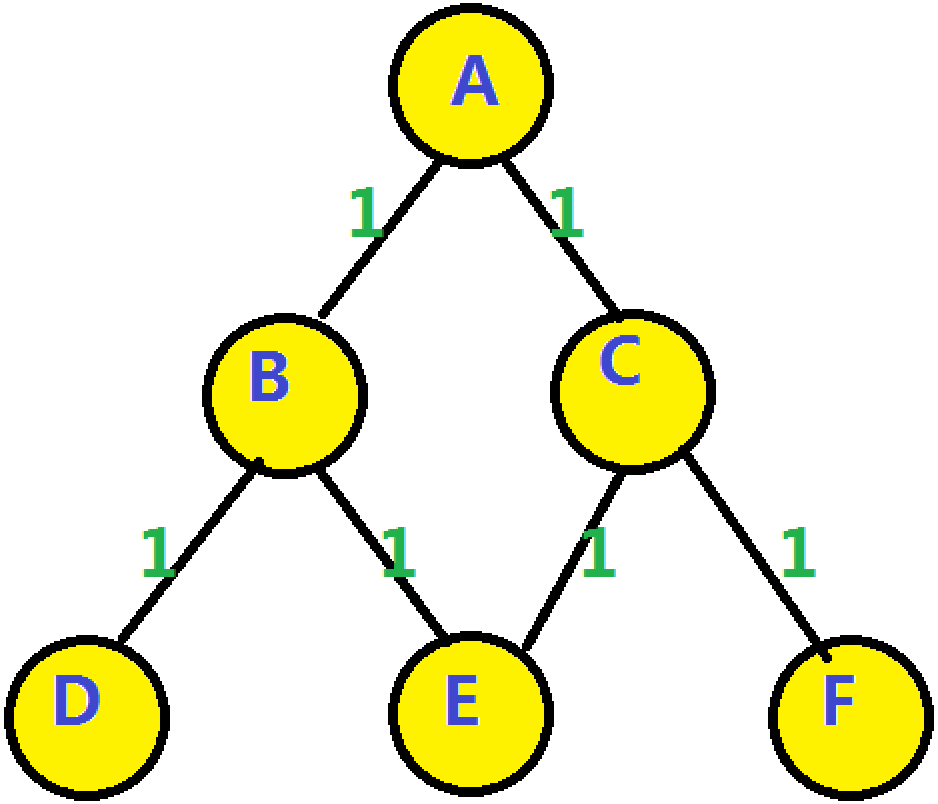


A SIMPLE IMPLEMENTATION

- Problem simplification:
- Graph G with |V| vertices and |E| edges
- Each edge e=(v1,v2) has weight 1

- BFS
- The time when you visit the vertex must be the distance to the source.
- D[i]

A SIMPLE IMPLEMENTATION



Visit order(BFS):

A, B, C, D, E, F

A SIMPLE IMPLEMENTAION

- The general form:
- Graph G with |V| vertices and |E| edges
- Each edge e=(vi,vj) has weight w(i,j) > 0

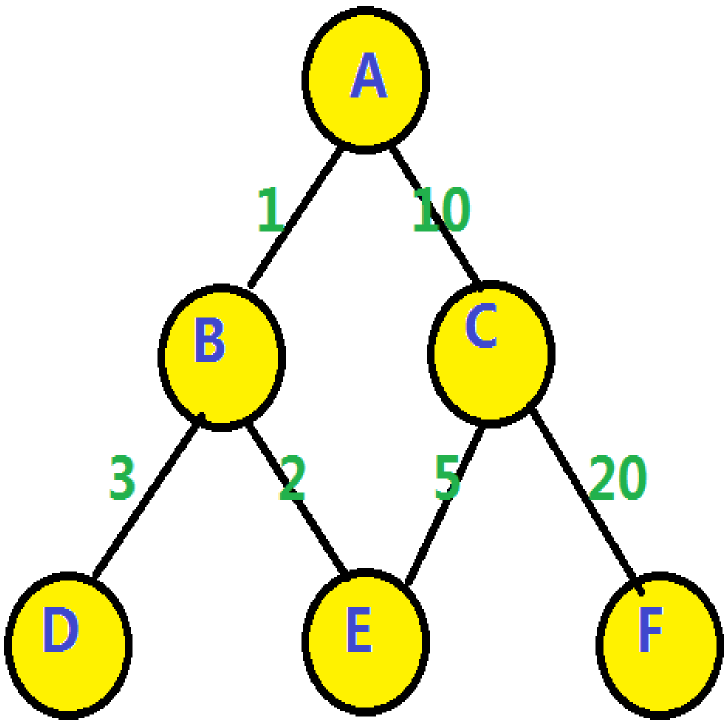
- How to compute D[i]
- Edge relaxation mechanism

EDGE RELAXATION

- Consider 2 vertices src and dst
- Edge e(src,dst) has weight w(src,dst)

- When to relax dst?
- D[src] + w(src,dst) < D[dst]
- Implement within BFS

A SIMPLE IMPLEMENTAION



Visit order:

A:D[A] = 0

B:D[B] = 1

C:D[C] = 10

D:D[D] = 1 + 3

E:D[E] = 1 + 2

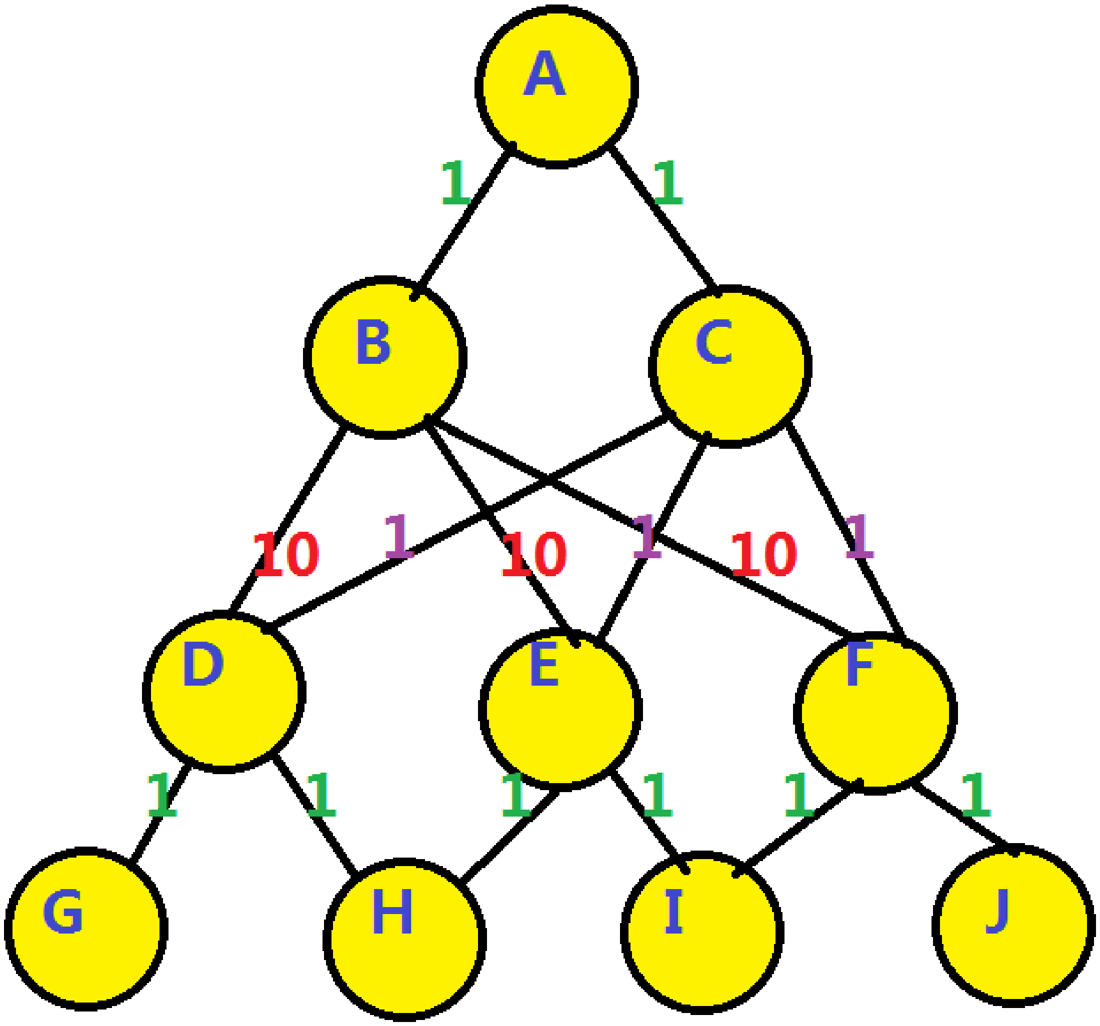
F:D[F] = 10 + 20

C:D[C] = 1 + 2 + 5

F:D[F] = 1 + 2 + 5 + 20

SPFA: Optimized BFS

- However, you may be disappoint with the time efficiency of BFS
- Look at the following example



- A,B,C,D,E,F,D,E,F,G,H,H,I,I,J,G,H,H,I,I,J

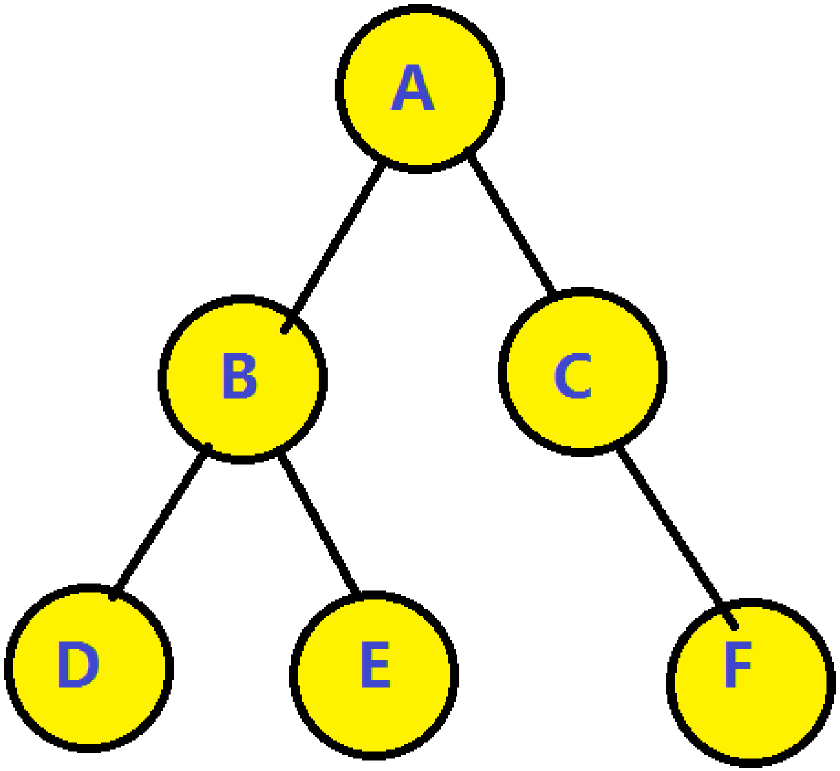
SPFA: OPTIMIZED BFS

- One more parameters for each point
- H(v)
- H(v) = 1 iff v is in the Queue
- The Algorithm:

```
If D[src] + w(src,dst) < D[dst] Then
Relax(dst)
If H[dst] = 0 Then Queue.add(dst)
Else do nothing
```

SHORTEST PATHS ON TREE

- **Characteristic of Tree?**



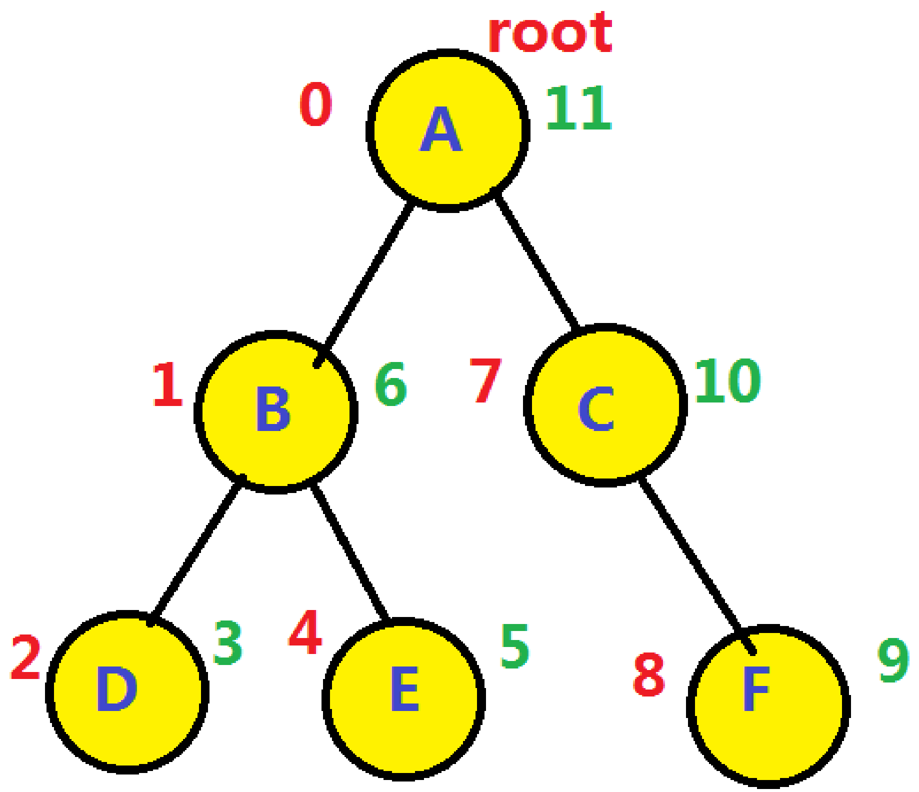
SHORTEST PATHS ON TREE

- Question?
- How to Find distance between 2 vertices on Tree?
- Time efficiency?

- Open Problem
- DFS-based implementation
- Given a rooted Tree, how to judge whether x is ancestor of y?

SHORTEST PATHS ON TREE

- In[v] & Out[v]



SHORTEST PATHS ON TREE

- DFS-based Implementation:
- Consider 2 vertices X and Y

```
If Ancestor(X) = Y Then
Dis(X,Y) = Dis(father(X),Y) +
  w(father(X),X)
Else
Dis(X,Y) = Dis(X,father(Y)) +
  w(father(Y),Y)
```

- Caution:

```
If X = Y Then Dis(X,X) = 0
```

THANK YOU FOR RESPECT

REFERENCES

- Princeton Online Course
- Algorithm Design