Algorithm Design X

Dynamic Programming II

Guoqiang Li
School of Software
SHANGHAI JIAO TONG UNIVERSITY

Shortest Reliable Paths

Shortest Reliable Paths

In a network, even if edge lengths faithfully reflect transmission delays, there may be other considerations involved in choosing a path.

Shortest Reliable Paths

In a network, even if edge lengths faithfully reflect transmission delays, there may be other considerations involved in choosing a path.

For instance, each extra edge in the path might be an extra "hop" fraught with uncertainties and dangers of packet loss.

Shortest Reliable Paths

In a network, even if edge lengths faithfully reflect transmission delays, there may be other considerations involved in choosing a path.

For instance, each extra edge in the path might be an extra “hop” fraught with uncertainties and dangers of packet loss.

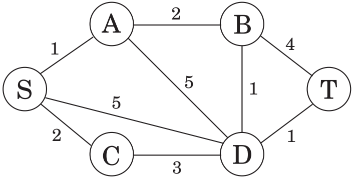
We would like to avoid paths with too many edges.

Shortest Reliable Paths

In a network, even if edge lengths faithfully reflect transmission delays, there may be other considerations involved in choosing a path.

For instance, each extra edge in the path might be an extra “hop” fraught with uncertainties and dangers of packet loss.

We would like to avoid paths with too many edges.



Shortest Reliable Paths

Suppose then that we are given a graph G with lengths on the edges, along with two nodes s and t and an integer k,

Shortest Reliable Paths

Shanghai Jiao Tong University

Suppose then that we are given a graph G with lengths on the edges, along with two nodes s and t and an integer k, and we want the shortest path from s to t that uses at most k edges.

Shortest Reliable Paths

Suppose then that we are given a graph G with lengths on the edges, along with two nodes s and t and an integer k, and we want the shortest path from s to t that uses at most k edges.

Dynamic programming will work!

Dynamic Programming

SHANGHAI JIAO TONG UNIVERSITY

Dynamic Programming

For each vertex v and each integer $i \leq k$, let

$$dist(v, i) = \text{ the length of the shortest path from } s \text{ to } v \text{ that uses } i \text{ edges}$$

Dynamic Programming

For each vertex v and each integer $i \leq k$, let

$$dist(v, i) = \text{ the length of the shortest path from } s \text{ to } v \text{ that uses } i \text{ edges}$$

The starting values $dist(v, 0)$ are ∞ for all vertices except s, for which it is 0.

Dynamic Programming

SHANGHAI JIAO TONG UNIVERSITY

For each vertex v and each integer $i \leq k$, let

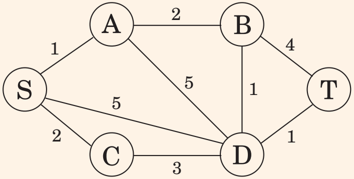
$$dist(v, i) = \text{ the length of the shortest path from } s \text{ to } v \text{ that uses } i \text{ edges}$$

The starting values $dist(v, 0)$ are ∞ for all vertices except s, for which it is 0.

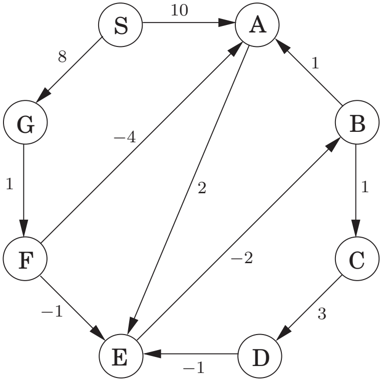
$$dist(v, i) = \min_{(u,v) \in E} \{dist(u, i-1) + l(u, v)\}$$

Shortest Reliable Paths

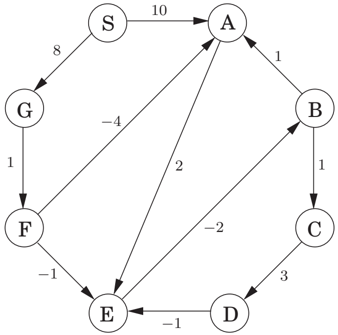
Find out the shortest reliable path from S to T, when $k = 3$.



An Example



Bellman-Ford Algorithm



	Iteration							
Node	0	1	2	3	4	5	6	7
S	0	0	0	0	0	0	0	0
A	∞	10	10	5	5	5	5	5
B	∞	∞	∞	10	6	5	5	5
C	∞	∞	∞	∞	11	7	6	6
D	∞	∞	∞	∞	∞	14	10	9
E	∞	∞	12	8	7	7	7	7
F	∞	∞	9	9	9	9	9	9
G	∞	8	8	8	8	8	8	8

All-Pairs Shortest Path

All-Pairs Shortest Path

What if we want to find the shortest path not just between s and t but between all pairs of vertices?

All-Pairs Shortest Path

What if we want to find the shortest path not just between s and t but between all pairs of vertices?

One approach would be to execute Bellman-Ford-Moore algorithm $|V|$ times, once for each starting node.

All-Pairs Shortest Path

What if we want to find the shortest path not just between s and t but between all pairs of vertices?

One approach would be to execute Bellman-Ford-Moore algorithm $|V|$ times, once for each starting node.

The total running time would then be $O(|V|^2|E|)$.

We'll now see a better alternative, the $O(|V|^3)$, named Floyd-Warshall algorithm.

Floyd-Warshall Algorithm

Dynamic programming again!

The Subproblems

Number the vertices in V as $\{1, 2, \ldots, n\}$,

The Subproblems

Number the vertices in V as $\{1, 2, \ldots, n\}$, and let

$dist(i, j, k) =$ the length of the shortest path from i to j in which only nodes $\{1, 2, \ldots, k\}$ can be used as intermediates.

The Subproblems

Number the vertices in V as $\{1, 2, \ldots, n\}$, and let

$dist(i, j, k) =$ the length of the shortest path from i to j in which only nodes $\{1, 2, \ldots, k\}$ can be used as intermediates.

Initially, $dist(i, j, 0)$ is the length of the direct edge between i and j, if it exists, and is ∞ otherwise.

The Subproblems

Number the vertices in V as $\{1, 2, \ldots, n\}$, and let

$dist(i, j, k) =$ the length of the shortest path from i to j in which only nodes $\{1, 2, \ldots, k\}$ can be used as intermediates.

Initially, $dist(i, j, 0)$ is the length of the direct edge between i and j, if it exists, and is ∞ otherwise.

For $k \geq 1$

$$dist(i, j, k) = \min\{dist(i, j, k-1), dist(i, k, k-1) + dist(k, j, k-1)\}$$

The Program


```
for i = 1 to n do
    for j = 1 to n do
        dist(i, j, 0) = ∞;
    end
end
for all (i, j) ∈ E do
    dist(i, j, 0) = l(i, j);
end
for k = 1 to n do
    for i = 1 to n do
        for j = 1 to n do
            dist(i, j, k) = min{dist(i, j, k − 1), dist(i, k, k − 1) + dist(k, j, k − 1)};
        end
    end
end
```

Traveling Salesman Problem

The Traveling Salesman Problem

A traveling salesman is getting ready for a big sales tour.

The Traveling Salesman Problem

A traveling salesman is getting ready for a big sales tour. Starting at his hometown, he will conduct a journey in which each of his target cities is visited exactly once before he returns home.

The Traveling Salesman Problem

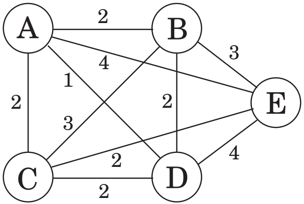
A traveling salesman is getting ready for a big sales tour. Starting at his hometown, he will conduct a journey in which each of his target cities is visited exactly once before he returns home.

Q: Given the pairwise distances between cities, what is the best order in which to visit them, so as to minimize the overall distance traveled?

A traveling salesman is getting ready for a big sales tour. Starting at his hometown, he will conduct a journey in which each of his target cities is visited exactly once before he returns home.

Q: Given the pairwise distances between cities, what is the best order in which to visit them, so as to minimize the overall distance traveled?

The brute-force approach is to evaluate every possible tour and return the best one. Since there are $(n-1)!$ possibilities, this strategy takes $O(n!)$ time.



The Traveling Salesman Problem

Denote the cities by $1, \ldots, n$, the salesman's hometown being 1,

The Traveling Salesman Problem

Denote the cities by $1, \ldots, n$, the salesman's hometown being 1, and let $D = (d_{ij})$ be the matrix of intercity distances.

The Traveling Salesman Problem

Denote the cities by $1, \ldots, n$, the salesman's hometown being 1, and let $D = (d_{ij})$ be the matrix of intercity distances.

The goal is to design a tour that starts and ends at 1, includes all other cities exactly once, and has minimum total length.

The Subproblems

For a subset of cities $S \subseteq \{1, 2, \ldots, n\}$ that includes 1, and $j \in S$, let $C(S, j)$ be the length of the shortest path visiting each node in S exactly once, starting at 1 and ending at j.

The Subproblems

For a subset of cities $S \subseteq \{1, 2, \ldots, n\}$ that includes 1, and $j \in S$, let $C(S, j)$ be the length of the shortest path visiting each node in S exactly once, starting at 1 and ending at j.

When $|S| > 1$, we define $C(S, 1) = \infty$.

The Subproblems

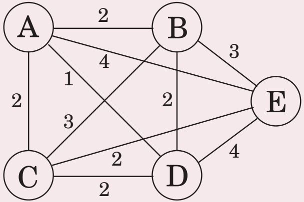
For a subset of cities $S \subseteq \{1, 2, \ldots, n\}$ that includes 1, and $j \in S$, let $C(S, j)$ be the length of the shortest path visiting each node in S exactly once, starting at 1 and ending at j.

When $|S| > 1$, we define $C(S, 1) = \infty$.

For $j \neq 1$ with $j \in S$ we have

$$C(S, j) = \min_{i \in S : i \neq j} C(S \backslash \{j\}, i) + d_{ij}$$

Exercise



The Program

$C(1,1) = 0$;
for $s = 2$ *to* n **do**
  for *all subsets* $S \subseteq \{1,2,\ldots,n\}$ **do**
    $C(S,1) = \infty$;
    for *all* $j \in S$ *and* $j \neq 1$ **do**
      $C(S,j) = \min_{i \in S: i \neq j} C(S \backslash \{j\}, i) + d_{ij}$;
    end
  end
end
`return`($\min_j C(\{1,2,\ldots,n\}, j) + d_{j1}$);

The Program

$C(1,1) = 0$;
for *$s = 2$ to n* **do**
  for *all subsets $S \subseteq \{1, 2, \ldots, n\}$* **do**
    $C(S,1) = \infty$;
    for *all $j \in S$ and $j \neq 1$* **do**
      $C(S,j) = \min_{i \in S: i \neq j} C(S \backslash \{j\}, i) + d_{ij}$;
    end
  end
end
`return`($\min_j C(\{1, 2, \ldots, n\}, j) + d_{j1}$);

There are at most $2^n \cdot n$ subproblems, and each one takes linear time.

The Program

$C(1,1) = 0$;
for $s = 2$ *to* n **do**
  for *all subsets* $S \subseteq \{1, 2, \ldots, n\}$ **do**
    $C(S,1) = \infty$;
    for *all* $j \in S$ *and* $j \neq 1$ **do**
      $C(S,j) = \min_{i \in S : i \neq j} C(S \setminus \{j\}, i) + d_{ij}$;
    end
  end
end
`return`($\min_j C(\{1, 2, \ldots, n\}, j) + d_{j1}$);

There are at most $2^n \cdot n$ subproblems, and each one takes linear time.

The total running time is therefore $O(n^2 \cdot 2^n)$.

Independent Sets in Trees

The Problem

A subset of nodes $S \subseteq V$ is an independent set of graph $G = (V, E)$ if there are no edges between them.

The Problem

A subset of nodes $S \subseteq V$ is an independent set of graph $G = (V, E)$ if there are no edges between them.

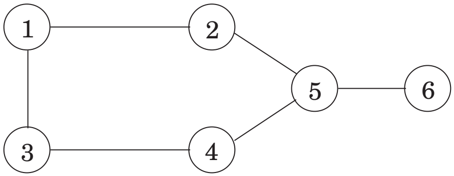
Finding the largest independent set in a graph is believed to be intractable.

The Problem

A subset of nodes $S \subseteq V$ is an independent set of graph $G = (V, E)$ if there are no edges between them.

Finding the largest independent set in a graph is believed to be intractable.

However, when the graph happens to be a tree, the problem can be solved in linear time, using dynamic programming.



The Subproblems

The Subproblems

SHANGHAI JIAO TONG UNIVERSITY

$I(u) =$ size of largest independent set of subtree hanging from u.

The Subproblems

$I(u)$ = size of largest independent set of subtree hanging from u.

$$I(u) = \max\{1 + \sum_{\text{grandchildren } w \text{ of } u} I(w), \sum_{\text{children } w \text{ of } u} I(w)\}$$

Homework

Homework

Assignment 4. Exercises 6.17, 6.20, 6.21, and 6.22.