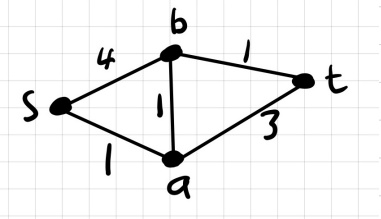
Shortest Paths in Edge Weighted Graphs

Recall: 

shortest path from s to t is $s\,a\,b\,t$ of length 3

Versions of the problem:

1. Given u, v find shortest uv path. —seems to involve solving (2)
2. Given u, find shortest uv path $\forall v$.
   "single source shortest path problem"
3. Find shortest uv path $\forall u, v$
   "all pairs shortest path problem" —dynamic programming: today (C)

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Recall: Negative weight cycles are trouble!

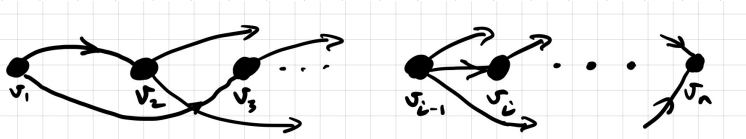
2. Single Source Shortest Paths in Directed Graphs

- no negative weights $O(m \log n)$
  Dijkstra's algorithm —last lecture
- no cycles $O(n + m)$ —today (A)
- general weights (but no negative cycle) $O(nm)$
  Bellman–Ford —today (B)

(A) Single source shortest paths in a directed acyclic graph (DAG)

$\equiv$ no directed cycle

Use topological sort $v_1 v_2 \cdots v_n$ so every edge (v_i, v_j) has $i < j$. recall DFS to find this



If v comes before s, there is no path $s \to v$.

So throw those vertices away. $s = v_1$

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
d_i = ∞ ∀ i
d_1 = 0
for i from 1 to n do
    for every edge (v_i, v_j) do
        if d_i + w(v_i, v_j) < d_j then
            d_j := d_i + w(v_i, v_j)
        fi
    od
od
```

$O(n + m)$

Claim This finds shortest paths.

Proof by induction on i.

Dynamic Programming for Shortest Paths in Graphs

Today we'll use dynamic programming for two problems:

(B) Single source shortest paths where edge weights may be negative but the graph has no negative weight cycle. Bellman–Ford. The original application of dynamic programming.

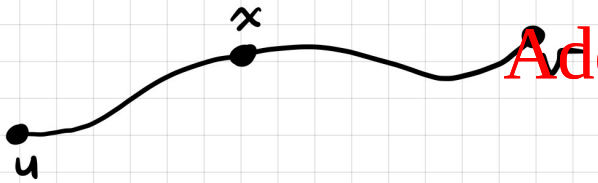
Note: If there is a negative weight cycle then shortest paths are not well-defined. Go around the cycle more and more to decrease length of path arbitrarily.

Assignment Project Exam Help

(C) All pairs shortest paths. Floyd–Warshall.

https://powcoder.com

Dynamic programming for shortest paths.



Add WeChat powcoder

If shortest uv path goes through x then it consists of shortest ux path + shortest xv path.

- we can try all x
- shortest ux and xv path are subproblems

How are these subproblems "smaller"?

Two possibilities:

1. They use fewer edges. This lead to a dynamic programming algorithm where we try paths of ≤ 1 edge, ≤ 2 edges, etc.
2. They don't use vertex x.

We will pursue: (1) for single source (B)
(2) for all pairs (C).

Single Source

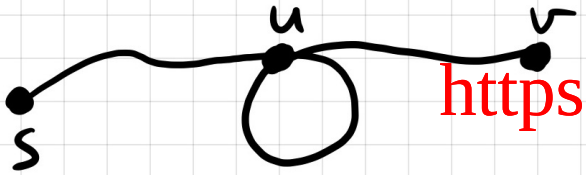
Let $d_i(v) =$ weight of shortest path from s to v using $\leq i$ edges.

$$\text{Then } d_1(v) = \begin{cases} 0 & \text{if } v = s \\ w(s,v) & \text{if } (s,v) \in E \\ \infty & \text{else} \end{cases}$$

And we want $d_{n-1}(v)$.

Why? Because a path with $\geq n$ edges would repeat a vertex giving a cycle.

Assignment Project Exam Help



https://powcoder.com

Add WeChat powcoder

Since every cycle has weight ≥ 0, removing the cycle is at least as good.

We compute d_i from d_{i-1}:

$$d_i(v) = \min \begin{cases} d_{i-1}(v) & (\text{use} \leq i-1 \text{ edges}) \\ \min_u(d_{i-1}(u) + w(u,v)) & (\text{use } i \text{ edges}) \end{cases}$$

Correctness We consider all possibilities for d_i. Then correct by induction on i.

Bellman–Ford

Initialize $d_1(v)$ for all v as above.
for i **from** 2 **to** $n-1$ **do**
  for $v \in V$ **do**
    $d_i(v) := d_{i-1}(v)$
    for each edge (u, v) **do**
      $d_i(v) := \min\{d_i(v), d_{i-1}(u) + w(u, v)\}$
    od
  od
od

Note: from v, we want edges directed in to v. $\star$

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Run time: $O(n \cdot (n + m))$

We can save space — re-use same $d(v)$.
Can also simplify code (and avoid issue $\star$)

$d(v) = \infty \ \forall \ v$
$d(s) = 0$
for i **from** 1 **to** $n-1$ **do**
  for $(u, v) \in E$ **do**
    $d(v) := \min\{d(v), d(u) + w(u, v)\}$
  od
od

Note the curious fact that i does not appear inside the loop.

Ex. See why the simplified code does the same as the original.
(In this form, it is more mysterious why the code works.)

In fact, we can exit from the top-level loop after an iteration in which no d value changes.
Ex. Justify this.

Can enhance the code to find actual shortest paths — just store parent pointer & update when d is updated. Allows to recover path $s \to v$ backwards from v.

Assignment Project Exam Help

Can also use this code to detect negative weight cycle reachable from s.

https://powcoder.com

How? Run 1 more iteration and see if any d value changes.
Ex. See why this works.

Add WeChat powcoder

Ex. Show how to detect negative weight cycle anywhere in the graph.
[Solution: add new s' and add edges (s', v) $\forall$ v, with weight 0.]

All pairs shortest paths

Given digraph G with edge weights $w : E \to \mathbb{R}$ (but no negative cycle) find shortest path from u to v $\forall\, u, v$. Can output the distances as an $n \times n$ matrix $D[u, v]$.

Repeated Bellman–Ford gives $O(n^2 m)$.

We'll use dynamic programming where intermediate paths use only a subset of the vertices.

Let $V = \{1, 2, \ldots, n\}$.
Let $D_i[u, v]$ = length of shortest uv path using intermediate vertices in $\{1, 2, \ldots, i\}$.

Assignment Project Exam Help

Solve subproblems $D_i[u, v]$ for all u, v as i goes from 0 to n.

$D_n[u, v]$ is what we really want.

https://powcoder.com

Initial info: $D_0[u, v] = \begin{cases} 0 & \text{if } u = v \\ w(u, v) & \text{if } (u, v) \in E \\ \infty & \text{otherwise} \end{cases}$

Add WeChat powcoder

Main recursive property $i > 0$:

$$D_i[u, v] = \min \begin{cases} D_{i-1}[u, i] + D_{i-1}[i, v] & \text{use vertex } i \\ D_{i-1}[u, v] & \text{don't} \end{cases}$$

Correctness this considers all possibilities for D_i. Then induction on i.

Floyd–Warshall Algorithm

```
Initialize D_0[u, v] as above
for i from 1 to n do
    for u from 1 to n do
        for v from 1 to n do
            D_i[u, v] := min{D_{i-1}[u, v], D_{i-1}[u, i] + D_{i-1}[i, v]}
        od
    od
od
```

Assignment Project Exam Help

Time: $O(n^3)$
Space: $O(n^3)$

https://powcoder.com

Exercise: Show that you can get $O(n^2)$ space by showing that the following works:

Add WeChat powcoder

```
for i from 1 to n do
    for u from 1 to n do
        for v from 1 to n do
            D[u, v] := min{D[u, v], D[u, i] + D[i, v]}
        od
    od
od
```

What if we want the actual path?

Along with $D[u, v]$, compute $\text{Next}[u, v] =$ the first vertex after u on a shortest u to v path.

If we update $D[u, v] = D[u, i] + D[i, v]$ then also update $\text{Next}[u, v] := \text{Next}[u, i]$.

Ex. Check how this works.

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Bellman explains the reasoning behind the term *dynamic programming* in his autobiography, *Eye of the Hurricane: An Autobiography:* (1984, page 159). He explains:

> I spent the Fall quarter (of 1950) at RAND. My first task was to find a name for multistage decision processes. An interesting question is, "Where did the name, dynamic programming, come from?" The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word research. I'm not using the term lightly; I'm using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term research in his presence. You can imagine how he felt, then, about the term mathematical. The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. What title, what name, could I choose? In the first place I was interested in planning, in decision making, in thinking. But planning, is not a good word for various reasons. I decided therefore to use the word "programming". I wanted to get across the idea that this was dynamic, this was multistage, this was time-varying. I thought, let's kill two birds with one stone. Let's take a word that has an absolutely precise meaning, namely dynamic, in the classical physical sense. It also has a very interesting property as an adjective, and that is it's impossible to use the word dynamic in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

From: https://en.wikipedia.org/wiki/Dynamic_programming