Dynamic Programming

Hengfeng Wei

hfwei@nju.edu.cn

June 11 – June 19, 2017

Dynamic Programming

3-D DP

Floyd-Warshall algorithm

(1) DP for Floyd-Warshall algorithm for APSP on directed graphs

Subproblem: dist$[i, j, k]$: the length of the shortest path from i to j via only nodes in $v_1 \cdots v_k$

Goal: dist$[i, j, n], \forall i, j$

3-D DP

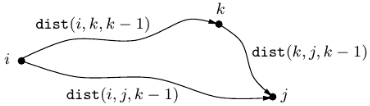
Floyd-Warshall algorithm

(1) DP for Floyd-Warshall algorithm for APSP on directed graphs

Make choice: Is v_k on the ShortestPath$[i, j, k]$?

Recurrence:

$$\text{dist}[i,j,k] = \min\{\text{dist}[i,j,k-1], \text{dist}[i,k,k-1] + \text{dist}[k,j,k-1]\}$$



3-D DP

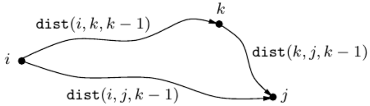
Floyd-Warshall algorithm

(1) DP for Floyd-Warshall algorithm for APSP on directed graphs

Make choice: Is v_k on the ShortestPath$[i, j, k]$?

Recurrence:

$$\text{dist}[i,j,k] = \min\{\text{dist}[i,j,k-1], \text{dist}[i,k,k-1] + \text{dist}[k,j,k-1]\}$$



Init:

$$\text{dist}[i,j,0] = \begin{cases} 0 & i = j \\ w(i,j) & (i,j) \in E \\ \infty & \text{o.w.} \end{cases}$$

3-D DP

Floyd-Warshall algorithm (Problem 6.25)

(2) Routing table for Floyd-Warshall algorithm

for all $k \leftarrow 1 \ldots n$ **do**
  for all $i \leftarrow 1 \ldots n$ **do**
    for all $j \leftarrow 1 \ldots n$ **do**
      if dist$[i, j] >$ dist$[i, k] +$ dist$[k, j]$ **then**
        dist$[i, j] \leftarrow$ dist$[i, k] +$ dist$[k, j]$

3-D DP

Floyd-Warshall algorithm (Problem 6.25)

(2) Routing table for Floyd-Warshall algorithm

for all $k \leftarrow 1 \dots n$ **do**
  for all $i \leftarrow 1 \dots n$ **do**
    for all $j \leftarrow 1 \dots n$ **do**
      if dist$[i, j] >$ dist$[i, k] +$ dist$[k, j]$ **then**
        dist$[i, j] \leftarrow$ dist$[i, k] +$ dist$[k, j]$

Time: $\Theta(n^3)$ Space: $\Theta(n^2)$

3-D DP

Floyd-Warshall algorithm (Problem 6.25)

(2) Routing table for Floyd-Warshall algorithm

for all $k \leftarrow 1 \ldots n$ **do**
  for all $i \leftarrow 1 \ldots n$ **do**
    for all $j \leftarrow 1 \ldots n$ **do**
      if $\text{dist}[i, j] > \text{dist}[i, k] + \text{dist}[k, j]$ **then**
        $\text{dist}[i, j] \leftarrow \text{dist}[i, k] + \text{dist}[k, j]$
        $\text{Go}[i, j] \leftarrow \text{Go}[i, k]$

Time: $\Theta(n^3)$ Space: $\Theta(n^2)$

3-D DP

Floyd-Warshall algorithm (Problem 6.25)

(2) Routing table for Floyd-Warshall algorithm

for all $i \leftarrow 1 \dots n$ **do**
  for all $j \leftarrow 1 \dots n$ **do**
    dist$[i, j] \leftarrow \infty$
    Go$[i, j] \leftarrow$ Nil
for all $(i, j) \in E$ **do**
  dist$[i, j] \leftarrow w(i, j)$
  Go$[i, j] \leftarrow j$
for all $i \leftarrow 1 \dots n$ **do**
  dist$[i, i] \leftarrow 0$
  Go$[i, i] \leftarrow$ Nil

3-D DP

Floyd-Warshall algorithm (Problem 6.25)

(2) Routing table for Floyd-Warshall algorithm

for all $i \leftarrow 1 \ldots n$ **do**
  for all $j \leftarrow 1 \ldots n$ **do**
    dist$[i, j] \leftarrow \infty$
    Go$[i, j] \leftarrow$ Nil
for all $(i, j) \in E$ **do**
  dist$[i, j] \leftarrow w(i, j)$
  Go$[i, j] \leftarrow j$
for all $i \leftarrow 1 \ldots n$ **do**
  dist$[i, i] \leftarrow 0$
  Go$[i, i] \leftarrow$ Nil

procedure $\text{PATH}(i, j)$
  if Go$[i, j] =$ Nil **then**
    Output "No Path."

  Output "i"
  while $i \neq j$ **do**
    $i \leftarrow$ Go$[i, j]$
  Output "i"

3-D DP

Floyd-Warshall algorithm (Problem 6.29)

(3) Find minimum-weighted cycle of directed graph $(w(e) > 0)$

$$\text{dist}[i, i] \leftarrow 0 \implies \text{dist}[i, i] \leftarrow \infty$$

$$\forall i : \text{dist}[i, i] = \infty$$

3-D DP

Floyd-Warshall algorithm (Problem 6.29)

(3) Find minimum-weighted cycle of directed graph $(w(e) > 0)$

$$\text{dist}[i, i] \leftarrow 0 \implies \text{dist}[i, i] \leftarrow \infty$$

$$\forall i : \text{dist}[i, i] = \infty$$

Q: $\exists e : w(e) < 0$

3-D DP

Floyd-Warshall algorithm (Problem 6.29)

(3) Find minimum-weighted cycle of directed graph $(w(e) > 0)$

$$\text{dist}[i,i] \leftarrow 0 \implies \text{dist}[i,i] \leftarrow \infty$$

$$\forall i : \text{dist}[i,i] = \infty$$

Q: $\exists e : w(e) < 0$

$\exists i : \text{dist}[i,i] < 0$

$\forall i : \text{dist}[i,i] \geq 0 \; (= \infty)$

Shortest paths on undirected graphs

Finding shortest paths in undirected graphs with possibly negative edge weights

2

The book "Algorithms" by Robert Sedgewick and Kevin Wayne hinted that (*see the quote below*) there are efficient algorithms for finding shortest paths in undirected graphs with possibly negative edge weights (***not*** by treating an undirected edge as two directed one which means that a single negative edge implies a negative cycle). However, no references are given in the book. Are you aware of any such algorithms?

2

> Q. How can we find shortest paths in undirected (edge-weighted) graphs?
>
> A. For positive edge weights, Dijkstra's algorithm does the job. We just build an EdgeWeightedDigraph corresponding to the given EdgeWeightedGraph (by adding two directed edges corresponding to each undirected edge, one in each direction) and then run Dijkstra's algorithm. ***If edge weights can be negative*** (***emphasis added***), efficient algorithms are available, but they are more complicated than the Bellman-Ford algorithm.

`algorithms` `graph-theory` `shortest-path` `weighted-graphs` `reference-question`

share cite edit close delete flag

edited Jun 9 at 14:15

asked Jun 9 at 13:58
hengxin
6,056 11 38

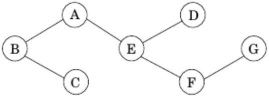
https://cs.stackexchange.com/q/76578/4911

Dynamic Programming

Minimum vertex cover on trees

Minimum vertex cover on trees [Problem: 2.2.18]

- Undirected tree $T = (V, E)$; No designated root!
- Compute (the size of) a minimum vertex cover of T



Minimum vertex cover on trees

Rooted T at any node r.

Minimum vertex cover on trees

Rooted T at any node r.

Subproblem: $I(u)$: the size of an MVC of subtree T_u rooted at u

Goal: $I(r)$

Minimum vertex cover on trees

Rooted T at any node r.

Subproblem: $I(u)$: the size of an MVC of subtree T_u rooted at u

Goal: $I(r)$

Make choice: Is u in MVC$[u]$?

Recurrence:

$$I(u) = \min\{\#\text{ children of } u + \sum_{v:\text{ grandchildren of } u} I(v), \quad 1 + \sum_{v:\text{ children of } u} I(v)\}$$

Minimum vertex cover on trees

Rooted T at any node r.

Subproblem: $I(u)$: the size of an MVC of subtree T_u rooted at u

Goal: $I(r)$

Make choice: Is u in MVC$[u]$?

Recurrence:

$$I(u) = \min\{\#\text{ children of } u + \sum_{v:\text{ grandchildren of } u} I(v), \\ 1 + \sum_{v:\text{ children of } u} I(v)\}$$

Init: $I(u) = 0$, if u is a leave

Minimum vertex cover on trees

DFS on T from root r:

when u is "finished":
if u is a leave **then**
$I(u) \leftarrow 0$
else
$I(u) \leftarrow \ldots$

Minimum vertex cover on trees

DFS on T from root r:

when u is "finished":
if u is a leave **then**
$I(u) \leftarrow 0$
else
$I(u) \leftarrow \ldots$

Greedy algorithm (Rough Proof!):

Theorem

There is an MVC which contains no leaves.

DP on DAG

Longest path in DAG (Problem 7.17)

- Direction: $\downarrow$ OR $\rightarrow$
- Score: $>=< 0$

DP on DAG

Longest path in DAG (Problem 7.17)

- Direction: $\downarrow$ OR $\rightarrow$
- Score: $>=<0$

1. digraph G
2. node weight $\rightarrow$ edge weight
3. adding an extra sink s
4. $G \rightarrow G^T$

DP on DAG

Longest path in DAG (Problem 7.17)

- Direction: $\downarrow$ OR $\rightarrow$
- Score: $> = < 0$

1. digraph G
2. node weight $\rightarrow$ edge weight
3. adding an extra sink s
4. $G \rightarrow G^T$

Compute a longest path from s in DAG

DP on DAG

Subproblem: dist$[v]$: longest distance from s to v

Goal: dist$[v], \forall v \in V$

DP on DAG

Subproblem: $\mathsf{dist}[v]$: longest distance from s to v

Goal: $\mathsf{dist}[v], \forall v \in V$

Make choice: What is the previous node before v on the longest path?

Recurrence:

$$\mathsf{dist}[v] = \max_{u \to v} \left(\mathsf{dist}[u] + w(u \to v) \right)$$

DP on DAG

Subproblem: $\text{dist}[v]$: longest distance from s to v

Goal: $\text{dist}[v], \forall v \in V$

Make choice: What is the previous node before v on the longest path?

Recurrence:

$$\text{dist}[v] = \max_{u \to v} \left(\text{dist}[u] + w(u \to v)\right)$$

Init: $\text{dist}[s] = 0$

DP on DAG

Subproblem: dist$[v]$: longest distance from s to v

Goal: dist$[v], \forall v \in V$

Make choice: What is the previous node before v on the longest path?

Recurrence:

$$\text{dist}[v] = \max_{u \to v} \left(\text{dist}[u] + w(u \to v) \right)$$

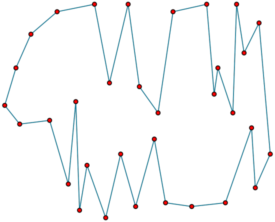
Init: dist$[s] = 0$

Compute dist$[v]$ in topo. order

Bitonic tour

Bitonic tour (Problem 7.18)

- Points: $P[1 \ldots n]$, $p_i = (x_i, y_i)$
- $x_1 < x_2 < \cdots < x_n$
- Bitonic tour: $p_1 \leadsto^{x_i < x_{i+1}} p_n \leadsto^{x_i > x_{i+1}} p_1$
- Compute a shortest bitonic tour.



Bitonic tour

$P_{i,j} (i \le j)$: bitonic path $p_i \leadsto^{x_i > x_{i+1}} p_1 \leadsto^{x_i < x_{i+1}} p_j$ includes all $p_1, p_2, \ldots, p_j$

Subproblem: $d[i, j]$: the length of a shortest bitonic path $P_{i,j}$

Goal: $d[n, n] = d[n-1, n] + l(p_{n-1} p_n)$

Bitonic tour

$P_{i,j} (i \leq j)$: bitonic path $p_i \leadsto^{x_i > x_{i+1}} p_1 \leadsto^{x_i < x_{i+1}} p_j$ includes all $p_1, p_2, \ldots, p_j$

Subproblem: $d[i,j]$: the length of a shortest bitonic path $P_{i,j}$

Goal: $d[n,n] = d[n-1,n] + l(p_{n-1}p_n)$

Make choice: Is p_{j-1} on the increasing path or the decreasing path?

Recurrence:

$$d[i,j] = d[i,j-1] + l(p_{j-1}p_j) \quad \forall i < j-1$$
$$d[i,j] = \min_{1 \leq k < j-1} \{d[k,j-1] + l(p_k p_j)\} \quad \forall i = j-1$$

Bitonic tour

$P_{i,j} (i \le j)$: bitonic path $p_i \rightsquigarrow^{x_i > x_{i+1}} p_1 \rightsquigarrow^{x_i < x_{i+1}} p_j$ includes all $p_1, p_2, \ldots, p_j$

Subproblem: $d[i,j]$: the length of a shortest bitonic path $P_{i,j}$

Goal: $d[n,n] = d[n-1,n] + l(p_{n-1}p_n)$

Make choice: Is p_{j-1} on the increasing path or the decreasing path?

Recurrence:

$$d[i,j] = d[i,j-1] + l(p_{j-1}p_j) \quad \forall i < j-1$$
$$d[i,j] = \min_{1 \le k < j-1} \{d[k,j-1] + l(p_k p_j)\} \quad \forall i = j-1$$

Init: $d[1,2] = l(p_1 p_2)$

Time:

$$O(n^2) = O(n \log n) + O(n^2) \cdot O(1) + O(n) \cdot O(n)$$

Dynamic Programming

The change-making problem

The change-making problem (Problem 7.12)

- Coins values: $x_1 \ldots x_n$
- Amount: v
- Is it possible to make change for v?

The change-making problem

The change-making problem (Problem 7.12(2), Problem 7.1 (Subset sum))

(2) Without repetition $(0/1)$

The change-making problem

The change-making problem (Problem 7.12(2), Problem 7.1 (Subset sum))

(2) Without repetition $(0/1)$

Subproblem: $C[i, w]$: Make change for w using only values of $x_1 \ldots x_i$?

Goal: $C[n, v]$

The change-making problem

The change-making problem (Problem 7.12(2), Problem 7.1 (Subset sum))

(2) Without repetition $(0/1)$

Subproblem: $C[i, w]$: Make change for w using only values of $x_1 \ldots x_i$?

Goal: $C[n, v]$

Make choice: Using value x_i or not?

Recurrence:

$$C[i, w] = C[i-1, w] \vee (C[i-1, w - x_i] \wedge w \geq x_i)$$

The change-making problem

The change-making problem (Problem 7.12(2), Problem 7.1 (Subset sum))

(2) Without repetition $(0/1)$

Subproblem: $C[i, w]$: Make change for w using only values of $x_1 \ldots x_i$?

Goal: $C[n, v]$

Make choice: Using value x_i or not?

Recurrence:

$$C[i, w] = C[i-1, w] \lor (C[i-1, w - x_i] \land w \geq x_i)$$

Init:

$$C[i, 0] = \text{true}$$
$$C[0, w] = \text{false, if } w > 0$$
$$C[0, 0] = \text{true}$$

Time: $O(nv)$

The change-making problem

The change-making problem (Problem 7.12(1))

(1) Unbounded repetition (∞)

The change-making problem

The change-making problem (Problem 7.12(1))

(1) Unbounded repetition (∞)

Subproblem: $C[i, w]$: Make change for w using only values of $x_1 \dots x_i$?

Goal: $C[n, v]$

The change-making problem

The change-making problem (Problem 7.12(1))

(1) Unbounded repetition (∞)

Subproblem: $C[i, w]$: Make change for w using only values of $x_1 \dots x_i$?

Goal: $C[n, v]$

Make choice: Using value x_i or not?

Recurrence:

$$C[i, w] = C[i-1, w] \lor (C[i, w - x_i] \land w \geq x_i)$$

The change-making problem

The change-making problem (Problem 7.12(1))

(1) Unbounded repetition (∞)

Subproblem: $C[i, w]$: Make change for w using only values of $x_1 \dots x_i$?

Goal: $C[n, v]$

Make choice: Using value x_i or not?

Recurrence:

$$C[i, w] = C[i-1, w] \vee (C[i, w - x_i] \wedge w \geq x_i)$$

Init:

$$C[i, 0] = \text{true}, \forall i = 0 \dots n$$
$$C[0, w] = \text{false, if } w > 0$$

Time: $O(nv)$

The change-making problem

The change-making problem (Problem 7.12(1))

(1) Unbounded repetition (∞)

The change-making problem

The change-making problem (Problem 7.12(1))

(1) Unbounded repetition (∞)

Subproblem: $C[w]$: Possible to make change for w?

Goal: $C[v]$

The change-making problem

The change-making problem (Problem 7.12(1))

(1) Unbounded repetition (∞)

Subproblem: $C[w]$: Possible to make change for w?

Goal: $C[v]$

Make choice: What is the first coin to use?

Recurrence:

$$C[w] = \bigvee_{i:\, x_i \le w} C[w - x_i]$$

The change-making problem

The change-making problem (Problem 7.12(1))

(1) Unbounded repetition (∞)

Subproblem: $C[w]$: Possible to make change for w?

Goal: $C[v]$

Make choice: What is the first coin to use?

Recurrence:

$$C[w] = \bigvee_{i:\, x_i \le w} C[w - x_i]$$

Init: $C[0] = \text{true}$

Time: $O(nv)$

The change-making problem

The change-making problem (Problem 7.12(1))

(1) Unbounded repetition (∞)

$$C[i, w] \textit{ vs. } C[w]$$

$$C[i, w] = C[i-1, w] \lor (C[i, w - x_i] \land w \geq x_i)$$

$$C[w] = \bigvee_{i:\, x_i \leq w} C[w - x_i]$$

The change-making problem

The change-making problem (Problem 7.12(3))

(3) Unbounded repetition with $\leq k$ coins

The change-making problem

The change-making problem (Problem 7.12(3))

(3) Unbounded repetition with $\leq k$ coins

Subproblem: $C[i, w, l]$: Possible to make change for w with $\leq l$ coins of values of $x_1 \ldots x_i$?

Goal: $C[n, v, k]$

The change-making problem

The change-making problem (Problem 7.12(3))

(3) Unbounded repetition with $\leq k$ coins

Subproblem: $C[i, w, l]$: Possible to make change for w with $\leq l$ coins of values of $x_1 \ldots x_i$?

Goal: $C[n, v, k]$

Make choice: Using value x_i or not?

Recurrence:

$$C[i, w, l] = C[i-1, w, l] \vee (C[i, w - x_i, l - 1] \wedge w \geq x_i)$$

The change-making problem

The change-making problem (Problem 7.12(3))

(3) Unbounded repetition with $\leq k$ coins

Subproblem: $C[i, w, l]$: Possible to make change for w with $\leq l$ coins of values of $x_1 \dots x_i$?

Goal: $C[n, v, k]$

Make choice: Using value x_i or not?

Recurrence:

$$C[i, w, l] = C[i-1, w, l] \vee (C[i, w - x_i, l-1] \wedge w \geq x_i)$$

Init:

$$C[0, 0, l] = \text{true}, \quad C[0, w, l] = \text{false, if } w > 0$$
$$C[i, 0, l] = \text{true}, \quad C[i, w, 0] = \text{false, if } w > 0$$

The change-making problem

The change-making problem (Problem 7.12(3))

(3) Unbounded repetition with $\leq k$ coins

The change-making problem

The change-making problem (Problem 7.12(3))

(3) Unbounded repetition with $\leq k$ coins

Subproblem: $C[w, l]$: Possible to make change for w with $\leq l$ coins?

Goal: $C[v, k]$

The change-making problem

The change-making problem (Problem 7.12(3))

(3) Unbounded repetition with $\leq k$ coins

Subproblem: $C[w, l]$: Possible to make change for w with $\leq l$ coins?

Goal: $C[v, k]$

Make choice: What is the first coin to use?

Recurrence:

$$C[w, l] = \bigvee_{i:\ x_i \leq w} C[w - x_i, k - 1]$$

The change-making problem

The change-making problem (Problem 7.12(3))

(3) Unbounded repetition with $\leq k$ coins

Subproblem: $C[w, l]$: Possible to make change for w with $\leq l$ coins?

Goal: $C[v, k]$

Make choice: What is the first coin to use?

Recurrence:

$$C[w, l] = \bigvee_{i:\ x_i \leq w} C[w - x_i, k - 1]$$

Init:

$$C[0, l] = \text{true},$$
$$C[w, 0] = \text{false, if } w > 0$$


THANK YOU