Part11-Dijkstra's shortest-path algorithm

Key videos	https://www.coursera.org/learn/algorithms-graphs-data-structures/lecture/rxrPa/dijkstras-shortest-path-algorithm https://www.coursera.org/learn/algorithms-graphs-data-structures/lecture/Pbpp9/dijkstras-algorithm-implementation-and-running-time
Note	
Period	@2020/04/25

Note

▼ Dijkstra's Shortest-Path Algorithm

- 问题描述：

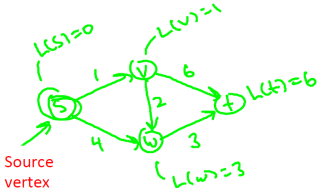
Input: directed graph G=(V, E). (m=|E|, n=|V|)
• each edge has non negative length l_e
• source vertex s

Output: for each $v \in V$, compute
L(v) := length of a shortest s-v path in G

Assumption:
1. [for convenience] $\forall v \in V, \exists s \Rightarrow v$ path
2. [important] $le \geq 0 \ \forall e \in E$

One of the following is the list of shortest-path distances for the nodes s,v,w,t, respectively. Which is it?

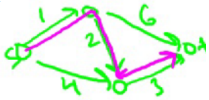
- ○ 0,1,2,3
- ○ 0,1,4,7
- ○ 0,1,4,6
- ○ 0,1,3,6



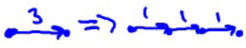
- 出现另外一种最短路径的原因：普通BFS计算路径是默认考虑编程为1

Why Another Shortest-Path Algorithm?

Question: doesn't BFS already compute shortest paths in linear time?
Answer: yes, IF $l_e = 1$ for every edge e.



Question: why not just replace each edge e by directed path of l_e unit length edges:
Answer: blows up graph too much



Solution: Dijkstra's shortest path algorithm.

Initialize:
- X = [s] [vertices processed so far]
- A[s] = 0 [computed shortest path distances]
- B[s] = empty path [computed shortest paths]

Main Loop
- while X≠V:

-need to grow x by one node

Main Loop cont'd:
- among all edges $(v, w) \in E$ with $v \in X, w \notin X$, pick the one that minimizes $A[v] + l_{vw}$

[call it (v*, w*)] Already computed in earlier iteratio

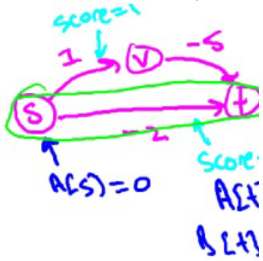
- add w* to X
- set $A[w^*] := A[v^*] + l_{v^*w^*}$
- set $B[w^*] := B[v^*]u(v^*, w^*)$

每次计算剩下为探索边组合到未探索点得最短路径

▼ Dijkstra's Algorithm: Examples

- Non-example:主要面向的是Nonnegative的边；与此相反，出现负边，并在基础上增加大常数，很有可能造成最短路径的转移

Question: why not reduce computing shortest paths with negative edge lengths to the same problem with non negative lengths? (by adding large constant to edge lengths)



Problem: doesn't preserve shortest paths !

Also: Dijkstra's algorithm incorrect on this graph !
(computes shortest s-t distance to be -2 rather than -4)

▼ Correctness of Dijkstra's Algorithm

▼ Dijkstra's Algorithm: Implementation and Running Time

- 习题：基础版本实现，会有(n-1)次循环，并且会粗暴的进行线性扫描linear scan

Which of the following running times seems to best describe a "naïve" implementation of Dijkstra's algorithm?

- ○ $\theta(m+n)$
- ○ $\theta(m \log n)$
- ○ $\theta(n^2)$
- ○ $\theta(mn)$

- (n-1) iterations of while loop
- $\theta(m)$ work per iteration

[$\theta(1)$ work per edge]

CAN WE DO BETTER?

- 使用全新数据结构：Heap(堆)，可以简化计算

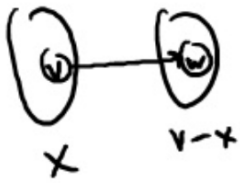
Heap Operations

Recall: raison d'être of heap = perform Insert, Extract-Min in O(log n) time.
[rest of video assumes familiarity with heaps]

Height ~ $\log_2 n$

- conceptually, a perfectly balanced binary tree
- Heap property: at every node, key <= children's keys
- extract-min by swapping up last leaf, bubbling down
- insert via bubbling up

You check: dominated by heap operations. (O(log(n)) each)
- (n-1) Extract mins
- each edge (v,w) triggers at most one Delete/Insert combo
(if v added to X first)



So: # of heap operations in O(n+m) = O(m)
So: running time = O(m log(n)) (like sorting)

Since graph is weakly connected

▼ Problem Set #2

1. Consider a directed graph with distinct and nonnegative edge lengths and a source vertex s. Fix a destination vertex t, and assume that the graph contains at least one s-t path. Which of the following statements are true? [Check all that apply.]

- [ ] There is a shortest s-t path with no repeated vertices (i.e., a "simple" or "loopless" such path).
- [ ] The shortest (i.e., minimum-length) s-t path might have as many as $n - 1$ edges, where n is the number of vertices.
- [ ] The shortest s-t path must include the minimum-length edge of G.
- [ ] The shortest s-t path must exclude the maximum-length edge of G.

AB~~C~~

C

This should not be selected

There are very small counterexamples (can you find one?).

1. Consider a directed graph with distinct and nonnegative edge lengths and a source vertex s. Fix a destination vertex t, and assume that the graph contains at least one s-t path. Which of the following statements are true? [Check all that apply.]

- [x] The shortest (i.e., minimum-length) s-t path might have as many as $n-1$ edges, where n is the number of vertices.

✓ **Correct**

- [ ] The shortest s-t path must include the minimum-length edge of G.
- [x] There is a shortest s-t path with no repeated vertices (i.e., a "simple" or "loopless" such path).

✓ **Correct**

- [ ] The shortest s-t path must exclude the maximum-length edge of G.

---

2. Consider a directed graph G with a source vertex s, a destination t, and nonnegative edge lengths. Under what conditions is the shortest s-t path guaranteed to be unique?

- ○ When all edge lengths are distinct positive integers.
- ◉ When all edge lengths are distinct powers of 2.
- ○ When all edges lengths are distinct positive integers and the graph G contains no directed cycles.
- ○ None of the other options are correct.

B

Correct

Two sums of distinct powers of two cannot be the same (imagine the numbers are written in binary).

3. Consider a directed graph $G = (V, E)$ and a source vertex s with the following properties: edges that leave the source vertex s have arbitrary (possibly negative) lengths; all other edge lengths are nonnegative; and there are no edges from any other vertex to the source s. Does Dijkstra's shortest-path algorithm correctly compute shortest-path distances (from s) in this graph?

- ○ Maybe, maybe not (depends on the graph)
- ○ Never
- ○ Only if we add the assumption that G contains no directed cycles with negative total weight.
- ◉ Always

D 相当于会把所有source出发的全部计算一遍

One approach is to see that the proof of correctness from the videos still works. A slicker solution is to notice that adding a positive constant MM to all edges incident to ss increases the length of every ss-vv path by exactly MM, and thus preserves the shortest path.

4. Consider a directed graph G and a source vertex s. Suppose G has some negative edge lengths but no negative cycles, meaning G does not have a directed cycle in which the sum of the edge lengths is negative. Suppose you run Dijkstra's algorithm on G (with source s). Which of the following statements are true? [Check all that apply.]

- [ ] Dijkstra's algorithm might loop forever.
- [x] Dijkstra's algorithm always terminates, but in some cases the paths it computes will not be the shortest paths from s to all other vertices.
- [x] Dijkstra's algorithm always terminates, and in some cases the paths it computes will be the correct shortest paths from s to all other vertices.
- [ ] It's impossible to run Dijkstra's algorithm on a graph with negative edge lengths.

BC

B

Correct

Nonnegativity of the edge lengths was used in the correctness proof for Dijkstra's algorithm; with negative edge lengths, the algorithm is no longer correct in general.

C

Correct

See Question 3.

5. Consider a directed graph G and a source vertex s. Suppose G contains a negative cycle (a directed cycle in which the sum of the edge lengths is negative) and also a path from s to this cycle. Suppose you run Dijkstra's algorithm on G (with source s). Which of the following statements are true? [Check all that apply.]

- [ ] Dijkstra's algorithm always terminates, but in some cases the paths it computes will not be the shortest paths from s to all other vertices.
- [x] Dijkstra's algorithm might loop forever.
- [x] It's impossible to run Dijkstra's algorithm on a graph with a negative cycle.
- [ ] Dijkstra's algorithm always terminates, and in some cases the paths it computes will be the correct shortest paths from s to all other vertices.

BC

B

This should not be selected

The algorithm always halts after n-1 iterations, where *n* is the number of vertices.

C

This should not be selected

Nothing about the description of the algorithm itself relies on there being no negative cycle.

5. Consider a directed graph G and a source vertex s. Suppose G contains a negative cycle (a directed cycle in which the sum of the edge lengths is negative) and also a path from s to this cycle. Suppose you run Dijkstra's algorithm on G (with source s). Which of the following statements are true? [Check all that apply.]

- [ ] Dijkstra's algorithm always terminates, and in some cases the paths it computes will be the correct shortest paths from s to all other vertices.
- [ ] It's impossible to run Dijkstra's algorithm on a graph with a negative cycle.
- [x] Dijkstra's algorithm always terminates, but in some cases the paths it computes will not be the shortest paths from s to all other vertices.

✓ **Correct**

- [ ] Dijkstra's algorithm might loop forever.

5. Consider a directed graph G and a source vertex s. Suppose G contains a negative cycle (a directed cycle in which the sum of the edge lengths is negative) and also a path from s to this cycle. Suppose you run Dijkstra's algorithm on G (with source s). Which of the following statements are true? [Check all that apply.]

- [ ] Dijkstra's algorithm always terminates, and in some cases the paths it computes will be the correct shortest paths from s to all other vertices.
- [ ] It's impossible to run Dijkstra's algorithm on a graph with a negative cycle.
- [x] Dijkstra's algorithm always terminates, but in some cases the paths it computes will not be the shortest paths from s to all other vertices.

✓ **Correct**

- [ ] Dijkstra's algorithm might loop forever.

Reference

- Dijkstra's algorithm 戴克斯特拉算法

Dijkstra's algorithm

Dijkstra's algorithm (or Dijkstra's Shortest Path First algorithm, SPF algorithm) is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in

𝕎 https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

戴克斯特拉算法

该算法存在很多变体：戴克斯特拉的原始版本仅适用于找到两个顶点之间的最短路径，后来更常见的变体固定了一个顶点作为源结点然后找到该顶点到图中所有其它结点的最短路径，产生一个 最短路径树 。 该算法解决了圖 上带权的单源最短路径问题[1] [10] :196-206。具体来

https://zh.wikipedia.org/wiki/%E6%88%B4%E5%85%8B%E6%96%AF%E7%89%B9%E6%8B%89%E7%AE%97%E6%B3%95

- Heap in Python

python数据结构之堆(heap)

本篇学习内容为堆的性质、python实现插入与删除操作、堆复杂度表、python内置方法生成堆。 区分堆(heap)与栈(stack)：堆与二叉树有关，像一堆金字塔型泥沙；而栈像一个直立垃圾桶，一列下来。 又被为优先队列(priority queue)。尽管名为优先队列，但堆并不是队列。回忆一下，在队列中，我们可以进行的限定操作是dequeue和enqueue。 ...

https://www.cnblogs.com/kumata/p/9201571.html

python数据结构之堆(heap)

本篇学习内容为堆的性质、python实现插入与删除操作、堆复杂度表、python内置方法生成堆。 区分堆(heap)与栈(stack)：堆与二叉树有关，像一堆金字塔型泥沙；而栈像一个直立垃圾桶，一列下来。 又被为优先队列(priority queue)。尽管名为优先队列，但堆并不是队列。回忆一下，在队列中，我们可以进行的限定操作是dequeue和enqueue。 ...

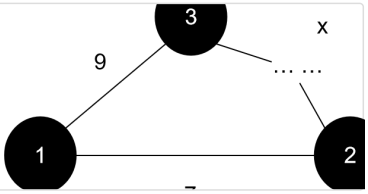
https://www.cnblogs.com/kumata/p/9201571.html

- Implememntation

python算法4.11--Dijkstra/迪杰斯特拉算法（无向图）_Python_DL$GIS-CSDN博客

单源最短路径算法，得到所有点的最短路径树 dijkstra为由路径树优化，得到最短路径树的搜索过程，这一过程的主要原则是最短枝优先，即搜索当前点到邻接点的最短节点（因此被认为是基于贪心策略）。基于这一原则可得到全局最短路径 以图为基础，建立最短路径集合set，为搜索点集合

https://blog.csdn.net/nominior/article/details/89217006



Dijkstra's Algorithm for Adjacency List Representation | Greedy Algo-8 - GeeksforGeeks

We recommend reading the following two posts as a prerequisite of this post. 1. Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm) 2. Graph and its representations We have discussed Dijkstra's algorithm and its implementation for adjacency matrix representation of graphs. The time complexity

https://www.geeksforgeeks.org/dijkstras-algorithm-for-adjacency-list-representation-greedy-algo-8/

Graph and its representations - GeeksforGeeks

Graph is a data structure that consists of following two components: 1. A finite set of vertices also called as nodes. 2. A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of a directed

https://www.geeksforgeeks.org/graph-and-its-representations/