问求算法设计与实践03

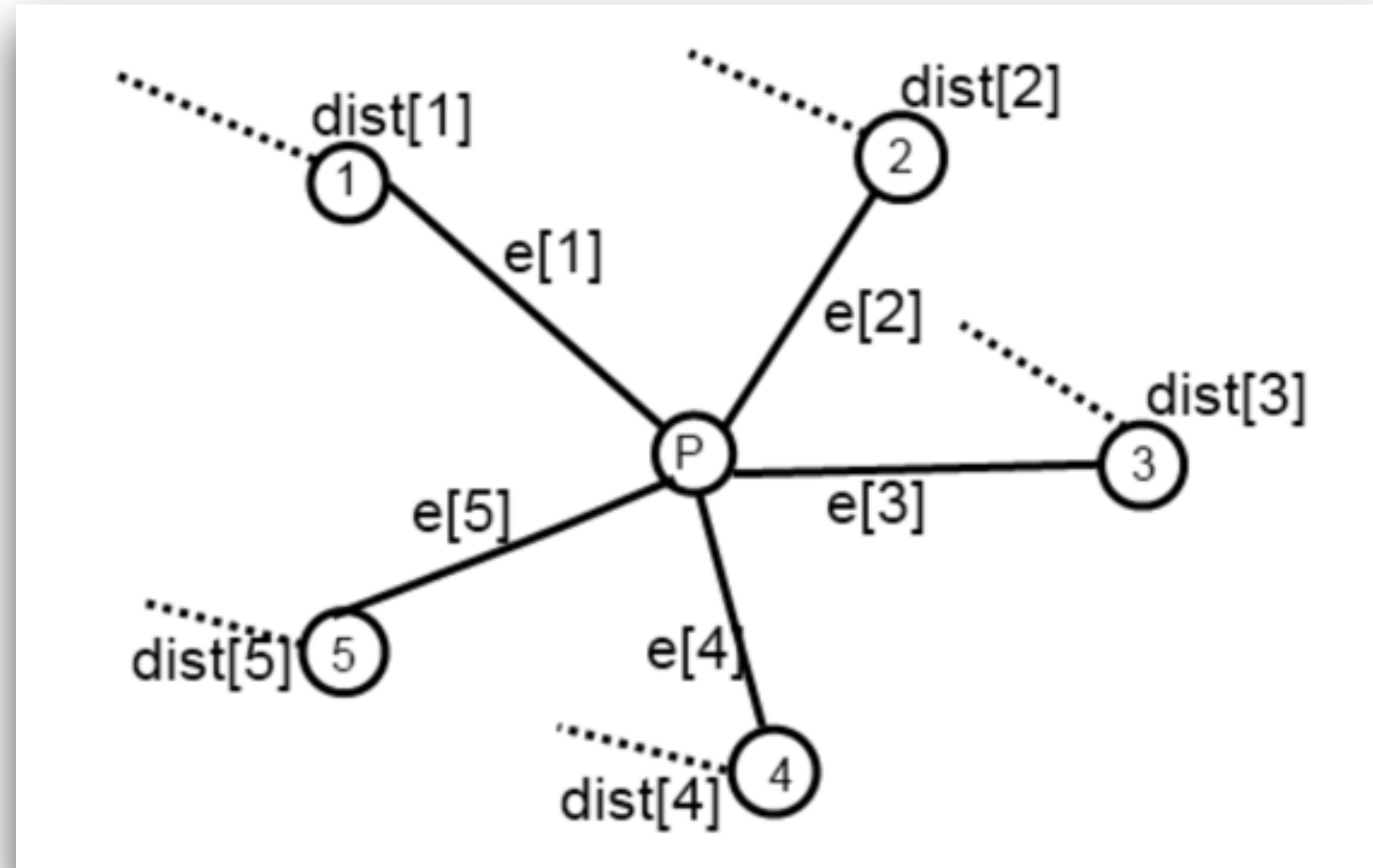
最短路

SSSP(Single-Source Shortest Path)

- 在一个带权图G上求出某一点u至其他所有点的最短路径

- 存在负环?

A simple idea



- dist[p]=min{dist[i]+weight[i][p]}
- 怎么确定次序?

如果不是DAG…

- 但图中存在一个环时…
- 递推陷入'死循环/递归'

- 怎么把环破开?
- 能否确定一个值出来

如果不是DAG...   cont'd

- 考虑到权值都是非负的
- 在一个环上，对于最短路径最短的那个点u,不可能通过环上的点来更新自己
- 确定下dist[u],这样环就破开了

如果不是DAG...   cont'd

- 不仅考虑环,推广到一般的图.
- **如果一个点u的最短路是当前最短的，那么他就不用再被其他点更新了**
- Order: 按照最短路大小依次更新递推式

Dijkstra Algorithm(1959)

- 初始: dist[start]=0 这是真实的最短路,也是目前最小的dist
- 每次拿出 dist值最小的 且 之前未拿出来过的 点u 去尝试更新其他点
- 1，拿出来的点的dist值单调不下降
- 2，dist[u]是真实最短路
- 数学归纳法

Time Complexity

- 每次找出最小的dist值O(N)，N次
- 每个点都会拿出来更新其他点,总的来算,就是边数O(M)
- O(N^2+M)

Time Complexity   cont’d

- 二叉堆
- 每次找出最小的dist值O(1)，N次
- 在二叉堆中删去最小的点O(logN), N次
- 每个点都会拿出来更新其他点,总的来算,就是边数O(M)
- 每次更新可能调整dist值, O(MlogN)
- O(N+MLogN)

Time Complexity cont'd

Operation	Binary[1]	Binomial[1]	Fibonacci[1]	Pairing[5]	Brodal[6][a]	Rank-pairing[8]	Strict Fibonacci[9]
find-min	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
delete-min	$\Theta(\log n)$	$\Theta(\log n)$	$O(\log n)$[b]	$O(\log n)$[b]	$O(\log n)$	$O(\log n)$[b]	$O(\log n)$
insert	$\Theta(\log n)$	$\Theta(1)$[b]	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
decrease-key	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(1)$[b]	$o(\log n)$[b][c]	$\Theta(1)$	$\Theta(1)$[b]	$\Theta(1)$
merge	$\Theta(m \log n)$[d]	$O(\log n)$[e]	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$

- O(M+NlogN)

负权图怎么办?

- 强行跑Dijkstra
- 或者所有边权加上一个常数使得为正 等等方法也是不行的

负权图怎么办?

- 强行跑Dijkstra
- 或者所有边权加上一个常数使得为正 等等方法也是不行的
- 回到最初的idea,一个图中当前最短路最短的点并不能保证是真实最短路(可能由另一个更远的点走一条负权边来更新)
- 换一个思路

Relax操作

- function Relax(edge E)
  - if (dist’[E.u]+E.weight<dist’[E.v])
    - dist’[E.v]=dist’[E.u]+E.weight
- End
- dist’[u]=0, if u=start , otherwise dist’[u]=Inf

Relax操作

- function Relax(edge E)
  - if (dist’[E.u]+E.weight<dist’[E.v])
    - dist’[E.v]=dist’[E.u]+E.weight
- End
- dist’[u]=0, if u=start , otherwise dist’[u]=Inf
- **一个图的dist’=dist当且仅当不能再进行松弛操作**

Relax的方式

- 每次rand()一条边Relax
- 有限,必定收敛
- 能处理负权图的问题?
- 可以

Bellman-Ford Algorithm(1958)

- 稍微别那么暴力..
- for i=1 to N do
  - for each edge E in graph G
    - Relax(E.u,E.v)
- O(NM)，简单能work

Bellman-Ford Algorithm cont’d

- WHY N?
- 其实这个算法还是有性质的...

Bellman-Ford Algorithm cont’d

- WHY N?
- 其实这个算法还是有性质的...
- 初始: 长度为0的最短路已经求出
- 第i轮更新(遍历所有边)后, 长度小于等于i的所有最短路都已求出
- 数学归纳法

Bellman-Ford Algorithm cont’d

- 可以用于检测负环
- N轮迭代结束后,再尝试Relax所有边
- 如果有Relax成功的,说明存在负环

- hint: 基于dfs的找负环方法更practical

Bellman-Ford Algorithm cont’d

- 一些优化
- 当一轮Relax没有更新任何dist值时就可以终止了(轮数=最长的最短路长度)
- 一个点之前如果上一轮没有被成功relax,这一轮也就不用考虑以他为起点的边了

SPFA(队列优化的Bellman-Ford)

- 用队列维护上一轮更新过dist值的点
- 初始队列只有起点
- 每次拿出队首, 更新与之相连的边，然后出队
  - 1，另一端点在队列中，更新dist值
  - 2，另一端点不在队列中, 更新dist值且加入队列
- 理论复杂度没有改变,实际效果非常practical

Floyd-Warshall Algorithm(1962)

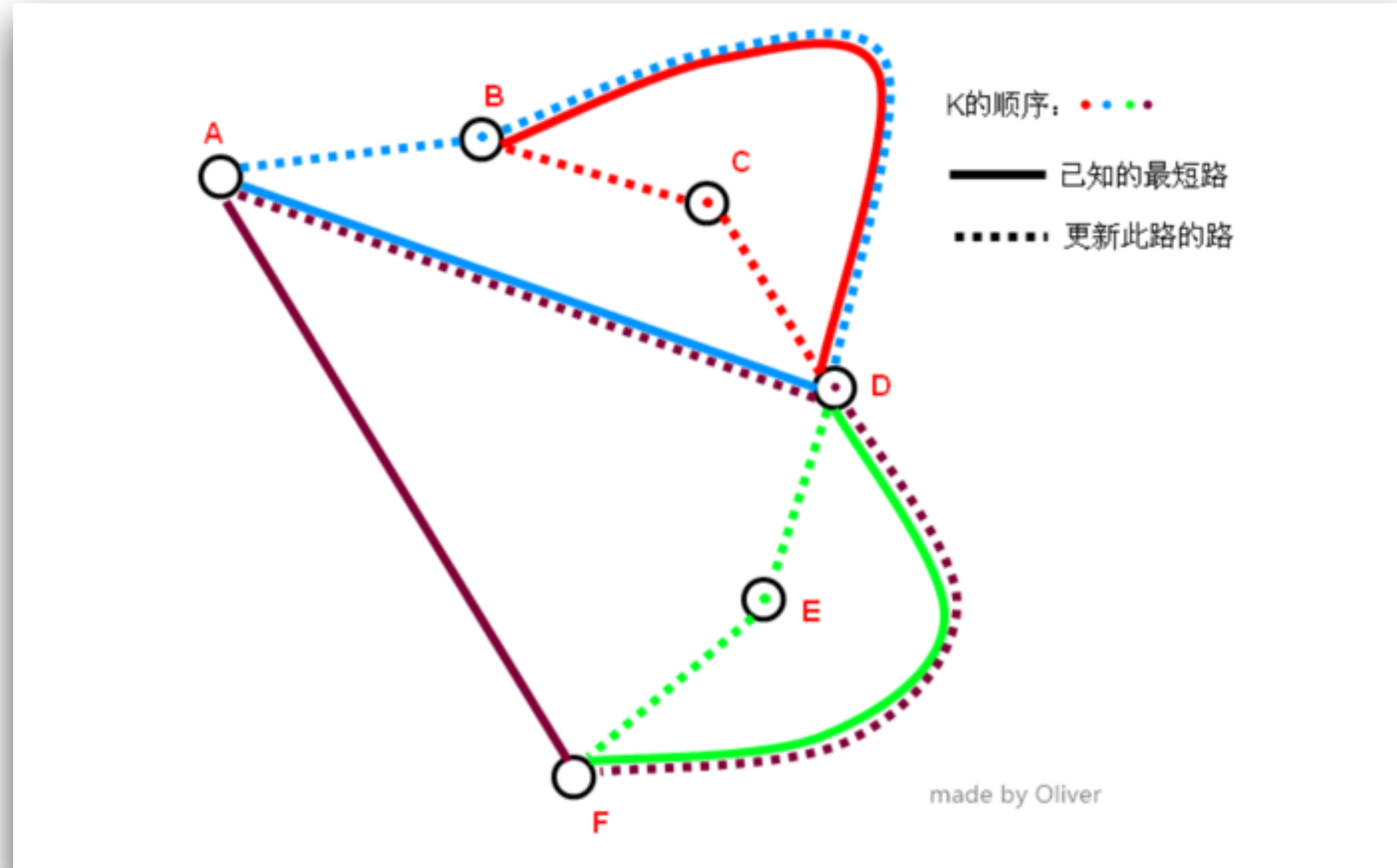
- 考虑另外一种划分’阶段’的方式
- F[i][j][k]表示i~j的 由顶点标号不超过k的节点 组成的最短路径长度
- F[i][j][k]=min(F[i][j][k-1],F[i][k][k-1]+F[k][j][k-1])

Floyd-Warshall Algorithm(1962)

```
let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
for each vertex v
   dist[v][v] ← 0
for each edge (u,v)
   dist[u][v] ← w(u,v)  // the weight of the edge (u,v)
for k from 1 to |V|
   for i from 1 to |V|
      for j from 1 to |V|
         if dist[i][j] > dist[i][k] + dist[k][j]
             dist[i][j] ← dist[i][k] + dist[k][j]
         end if
```

- O(N^3)
- 可以运行在负权图，可以检测负环

Floyd-Warshall Algorithm(1962)



Problem 2

Problem 2. 给出一张 N 个点的无向连通图，求 S 到 E 经过 K 条边的最短路。

$N \leq 100, K \leq 10^6$

Problem 2　cont’d

- 令G为一个N*N的矩阵, G[i][j]=1表示有i->j的边，否则为0
- G^2=G*G = ?
- G^2[i][j]=sigma G[i][k]*G[k][j]

Problem 2　cont’d

- 令G为一个N*N的矩阵, G[i][j]=1表示有i->j的边，否则为0
- G^2=G*G = ?
- G^2[i][j]=sigma G[i][k]*G[k][j]
- G^K 表示i->j走K条边有多少种走法
- 把乘法换成取min即为最短路
- 利用快速幂, O(N^3logK)

problem 3

Problem 3. 有一个 N 个点 M 条边的带权有向图，每条边权值范围 $1\sim N$. 求点 1 到点 N 的最短路.

$N, M \le 10^6$

- 修订:最短路长度小于N

Problem 3　cont’d

- 直接最短路当然可以，能否有更好的时间复杂度?
- 权值范围很小(1~N)
- 回顾dijkstra要求一个数据结构, 能查找最小值，删掉最小值，调整某个元素的值
- 而且最小值是单调不下降的

Problem 3   cont’d

- 我们维护N+1个链表头, L[0],L[1],…L[N] 和一个指针head指向0
- 初始将起点插入链表L[0]中
- 每次将L[head]链表中的元素取出, 更新相邻的点

Problem 3 cont’d

- 我们维护N+1个链表头, L[0],L[1],...L[N] 和一个指针head指向0
- 初始将起点插入链表L[0]中
- 每次将L[head]链表中的元素取出, 更新相邻的点
- 每次减小dist值时，O(1)从原链表中删除, O(1)插入新的链表中去
- head++
- O(N+M)

Problem 4

Problem 4. 有一个 N 个点 M 条边的带权有向图，你可以将一条边缩短一半.
求缩短后点 *1* 到点 N 的最短路径长度最少是多少。
$N, M \le 10^5$

- 边权为正

Problem 4   cont’d

- 一种方法是...
- 求两遍最短路,一次以1为起点，一次以N为起点(将边反向)
- 枚举每条边(i,j)， 尝试缩小一半,然后加上1~i和j~N的最短路
- 1~i和j~N的最短路相交?

Problem 4　cont’d

- 另一种更general的方式是**拆点**
- G[i][0]表示第i个点，还没有使用过减半
- G[i][1]表示第i个点，已经使用过减半

Problem 4　cont’d

- 另一种更general的方式是**拆点**
- G[i][0]表示第i个点，还没有使用过减半
- G[i][1]表示第i个点，已经使用过减半
- 对于边(i,j)
  - G[i][1]连向G[j][1]
  - G[i][0]连向G[j][0]，G[j][1](边权减半)
- 在新图G上运行最短路算法即可