Chapter0. Introduction

2019年2月20日 18:59

Note Taking Area

Course information

- Textbook: Algorithm design.
- Grades:
  - Final exam 40%
  - Lab 30%
  - Homework 20%
  - Attendance 10% - Be aware that sign in will be carried in lecture.
- Office hours: Wednesday 14:00 - 16:30, room 1011 A7 building, Zhiyuan.
- The lecture teacher has heavy accent.

Cue Column

Summaries

1. Course information.

Chapter1. Some Representative Problems: G-S

2019年2月20日 19:28

Note Taking Area

Stable matching

- 雇主与雇员的两条优先队列，满足稳定情况问题。
  - 稳定情况（其中之一成立）：
    i. 雇主E对他所接受的每个雇员都对该雇员A感兴趣；
    ii. 雇员A对他目前工作状况比雇主E感兴趣。
  - Unstable condition (both hold):

    Unstable pair: applicant x and hospital y are unstable if:
    - x prefers y to its assigned hospital.
    - y prefers x to one of its admitted students.
- **Stable assignment**: assignment with no unstable pair.
- Simplify into base case: one to one match, and all in match.
  - We'll consider the marriage as example in the following section.

Gale-Shapley algorithm / propose-and-reject algorithm

- The description of steps of algorithm:
  a. A male M make a proposal to highest female W, they are in date.
  b. Then every step the male A whole hasn't in date make a proposal to a highest priority female whole hasn't been made a proposal by A, and female selects the males and choose the highest priority (the other one isn't in date again).
  c. Until all males and females are in date, the algorithm done.
- The pseudo code:

初始所有的 $m \in M$ 和 $w \in W$ 都是自由的
While 存在男人 m 是自由的且还没对每个女人都求过婚
  选择这样一个男人 m
  令 w 是 m 的优先表中 m 还没求过婚的最高排名的女人
  If w 是自由的 then
    (m,w) 变成约会状态
  Else w 当前与 m' 约会
    If w 是更偏爱 m' 而不爱 m then
      m 保持自由
    Else w 更偏爱 m 而不爱 m',
      (m,w) 变成约会状态
      m' 变成自由
    Endif
  Endif
Endwhile
输出已约会对的集合 S.

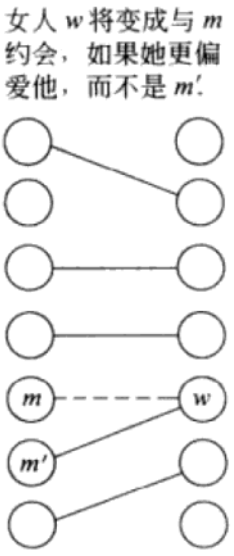

图 1.2 当自由男人 m 向女人 w 求婚时，G-S 算法的一个中间状态

Cue Column

Analysis and proof of G-S algorithm

- Female ω keeps engagement since her first proposal, and her couple becomes better.
- 男性m求过婚的女性在优先表中越来越差。
- G-S算法至多在n^2次迭代之后终止。

  证　正如我们这里打算做的，一个给出算法运行时间上界的有用策略是找出进展的度量标准. 即我们找出某种精确的方法说这个算法执行的每一步使得它离终止更接近.

  在当前这个算法的情况下，每次迭代由某个男人向一个以前没有求过婚的女人(唯一的一次)求婚构成. 于是，如果令 $P(t)$表示到迭代 t 结束 m 已经向 w 求过婚的那些(m,w)对的集合，我们看到，对所有的 t，$P(t+1)$的大小严格大于 $P(t)$的大小. 但是总计只存在 n^2 个可能的男人和女人的对，因此 $P(\cdot)$的值在算法的整个进程中至多可能增加 n^2 次. 从而得到至多可能存在 n^2 次迭代. ■
- 如果m在算法执行的某时刻是自由的，那么存在一个他还没有求婚的女性。

  证　假设存在一个点，此刻 m 是自由的，但是已经向每个女人都求过婚了. 那么根据命题 1.1，n 个女人中的每个人在这个时间点上是约会状态. 由于约会的对的集合构成一个匹配，在这个时间点上一定也有 n 个约会的男人. 但是，总共只有 n 个男人，且 m 不在约会，因此这是矛盾. ■
- 终止时返回的集合S是一个完美匹配。
- G-S算法的一次执行，返回一个稳定匹配集合S。
- 所有的执行得到同样的匹配，G-S算法的每次执行都得到集合S*，其中S*表示一个对的集合。
- 在稳定匹配S*中每位女性都与她最差的有效伴侣配对。

Lesson submission and score

- The online lab submission: 30%
  - Only one submission one week, which contains 10 test cases and each test case has 10 points.
  - The first submit will get full score, and one wrong answer will deduct 5 points.
- The theorical homework: 20%
  - Which will be submitted on sakai, at irregular intervals.
- Plagiarism policy:
  - If you use online code, write comment of original website.
- Code standards:
  - 《阿里巴巴Java开发手册 1.3.1》
  - 《Google style guide》
    - https://google.github.io/styleguide/

Summaries

1. Stable matching.
2. Gale-Shapley algorithm.

Chapter3. 图

2019年3月13日 19:41

NOTE TAKING AREA

基本定义与应用

- **命题3.1** 每棵**n个节点的树**恰好**有n-1条边。**
- **定理3.2** 设G是具有**n个节点的无向图**，下面**任意两个语句**都可以**推出第三个语句**：
  - G是**连通**的。
  - G**不包含一个圈。**
  - G有**n-1**条边。

宽度优先搜索

- 连通性问题（s-t连通性的确定问题）：在G中是否存在一条从s到t的路径?
- BFS算法构造的层：
  - 层L_1由s的所有邻居组成（有时也用L_0表示）。
  - 对于未定义的第一个层L_{j+1}由不属于前面的层并且有一条边通向层L_j的某个节点的所有节点组成。
- **定理3.3** 对每个j≥1，由BFS产生的层L_j恰好由所有到s距离为j的节点组成，存在一条从s到t的路径当且仅当t出现在某个层中。
- **定理3.4** 设T是一棵宽度优先搜索树，设x和y是T中分别属于层L_i和L_j的节点，并且设(x,y)是G的一条边，那么i与j至多差1。
- 使用BFS确定连通分支：

  R 将由那些有一条路径从 s 通向它的结点组成
  初始 $R=\{s\}$
  While 存在一条边 (u,v)，其中 $u\in R$ 而 $v\notin R$
  　将 v 加到 R 中
  Endwhile

  - 算法结束时产生的集合R恰好是G的包含s的连通分支。

深度优先搜索

- 深度优先搜索（DFS）的伪代码：

  DFS(u)：
  　将 u 标记为"Explored"并且把 u 加到 R
  　For 每条邻接到 u 的边 (u,v)
  　　If v 没有标记为"Explored"then
  　　递归调用 DFS(v)
  　　Endif
  　Endfor

- 命题和定理：
  - **命题3.6** 对于给定的递归调用DFS(u)，在这次激活和这个递归调用结束之间被标记为"Exprored"的所有的节点都是u在T中的后代。
  - **定理3.7** 设T是一棵深度优先搜索树，x与y是T中的节点，且(x,y)是G中不属于T的一条边，那么x或y之中一个是另一个的祖先。

- **定理3.8** 对图中任两个节点s与t，它们的连通分支或者相等，或者不相交。

用优先队列与栈实现图的遍历

- 图的表示：邻接表和邻接矩阵。
  - **命题3.9** $O(\sum_{v\in V} n_v) = 2m.$
    - 其中m为与v相交的边数，Adj[v]的表长是n_v。
  - **定理3.10** 一个图的邻接矩阵表示需要$O(n^2)$的空间，而邻接表表示只需要O(m+n)的空间。
- 宽度优先搜索的实现：

```
BFS(s)：
  置 Discovered[s]=true 且对所有其他的 v，置 Discovered[v]=false
  初始化 L[0]由单个元素 s 构成
  置层计数器 i=0
  置当前的 BFS 树 T=∅
  While L[i]不空
    初始化一个空表 L[i+1]
    For 每个结点 u∈L[i]
      考虑每条关联到 u 的边(u,v)
      If Discovered[v]=false then
        置 Discovered[v]=true
        把边(u,v)加到树 T 上
        把 v 加到表 L[i+1]
      Endif
    Endfor
    把层计数器加 1
  Endwhile
```

  - **定理3.11** 如果图是由邻接表给出的，BFS算法的上述实现将以O(m+n)时间复杂度运行。
- 深度优先搜索的实现

```
DFS(s)：
  初始化 S 为具有一个元素 s 的栈
  While S 非空
    从 S 中取一个结点 u
    If Explored[u]=false then
      置 Explored[u]=true
      For 每条与 u 关联的边(u,v)
        把 v 加到栈 S
      Endfor
    Endif
  Endwhile
```

  - 考虑将Explored标记放置于元素入栈之前，以节约内存占用。
  - **定理3.12** 上述算法在下面的意义上实现DFS，它按照与上一节递归的DFS过程恰好相同的次序（除了每个邻接表是按照相反的次序处理之外）访问节点。
  - **定理3.13** 如果图是由邻接表给出的，DFS算法的实现将以O(m+n)时间运行。

二分性测试：宽度优先搜索的应用

- 二部图：对结点着红色和蓝色，使得每条边有一个红端点和一个蓝端点。

- **定理3.14** 如果一个图是二部图，那么它不可能包含一个奇圈。
- **定理3.15** 设G是一个连通图，令L_1，L_2，...是由始于结点s的BFS所产生的层，那么下面两件事一定恰好成立其一：
    - G中没有边与同一层的两个节点相交，在这种情况下G是二部图，其中偶数层的节点可以着红色，奇数层的节点可以着蓝色。
    - G中有一条边与第一层的两个节点相交。在这种情况下，G包含一个奇数长度的圈，且因此不可能是二部图。

有向图中的连通性

- **命题3.16** 如果u和v是相互可达的，v和w是相互可达的，那么u和w也是相互可达的。
- **定理3.17** 对有向图中的任何两个节点s和t，它们的强连通分支或者相等，或者不相交。

DAG与拓扑排序的内容省略，请参考DSAA相关内容。

CUE COLUMN

SUMMARIES

1. 图的基本定义和应用。
2. 宽度优先搜索。
3. 广度优先搜索。
4. 用队列和栈实现图的遍历。
5. 二分性测试：宽度优先搜索的应用。
6. 有向图的连通性。

Chapter4. 贪心算法

2019年3月20日　14:18

NOTE TAKING AREA

贪心算法

- 定义：算法通过一些**小的步骤**来建立一个解，在每一步根据**局部情况**选择一个决定使得某些**主要的指标能得到优化。**
- 贪心算法领先：贪心算法的进展**每一步**都比其他算法好。
- 交换论证：考虑对某问题的任何可能的解，逐渐把它**转换成由贪心算法找到的解而不损害它的质量。**
- 例题：图中的最短路径、最小生成树问题、Huffman码，最小费用有向树问题。

区间调度：贪心算法领先

- 一点一个需求i_1被接受，则拒绝所有与i_1不相容的需求，选择下一个被接收的需求i_2，并且拒绝与i_2不相容的需求，直到遍历完所有需求。

```
初始令 R 是所有需求的集合，设 A 为空
While R 不空
    选择一个有最小结束时间的需求 i∈R
    把 i 加到 A 中
    从 R 中删除与需求 i 不相容的所有需求
Endwhile
返回集合 A 作为被接受的需求集合
```

- 分析算法：首先，算法返回的集合A中的区间都是相容的。
  - **命题4.1** A是一个相容的需求集
  - **命题4.2** 对所有的指标$r \le k$，我们有$f(i_r) \le f(j_r)$。
  - **命题4.3** 贪心算法返回一个最优的集合A。

    证　我们现在用反证法证明这个论断. 如果 A 不是最优的，那么一个最优集合 O 一定有更多的需求，即我们有一定有 $m>k$. 对 $r=k$ 应用命题 4.2，我们得到 $f(i_k) \leqslant f(j_k)$. 因为 $m>k$，在 O 中存在一个需求 j_{k+1}. 这个需求在 j_k 结束之后开始，因此也在 i_k 结束之后开始. 所以在删除了所有与需求 $i_1, i_2, \cdots, i_k$ 不相容的需求之后，可能的需求集合 R 仍旧包含 j_{k+1}. 但是贪心算法 i_k 停止在 i_k，而假设它仅当 R 为空时停止，产生矛盾. ■
- 实现与运行时间：算法运行时间为O(nlogn)。
- 相关问题：调度所有的区间

  问题　在区间调度问题中，存在一种单一的资源和以时间区间形式表示的许多需求，因此我们必须选择哪些需求被接受且哪些需求被拒绝. 如果我们有许多相同的资源可用，而且我们想使用尽可能少的资源安排所有的需求，那么就产生一个相关的问题. 由于这里的目标是把所有的区间划分到多个资源，我们将把它看做一个区间划分问题.

  例如，假设每个需求与一个在特定时间区间教室里需要安排的课对应. 我们想使用尽可能少的教室满足所有这些需求. 因而在我们的安排下教室是多个资源，并且基本的约束是两个在时间上重叠的课必须被安排在不同的教室. 采用等价的说法，区间需求也可以是需要在特定时间区间被处理的工作，而资源可能是能够处理这些工作的机器.
  - **命题4.4** 在任何区间划分的实例中，资源数必须至少是区间集合的深度。
  - 设计算法：设d是区间集合的深度，显示怎样对每个区间分配一个*标签*，即数集{1,

2, ..., d}，重叠的区间用不同的数做标签（名字对于数字的映射）。

```
任意打破区间的并列，根据开始时间对它们排序
设 I_1, I_2, ..., I_n 表示按这个次序排列的区间
For j=1,2,3,...,n
  For 每个按照上述次序领先于 I_j 并且与它重叠的区间 I_i
    从对 I_j 的考虑中排除 I_i 的标签
  Endfor
  If 在{1,2,...,d}中存在任何还没有被排除的标签 then
    分配一个没排除的标签给 I_j
  Else
    保留 I_j 不分配标签
  Endif
Endfor
```

- **命题4.5** 如果我们使用上述贪心算法，每个区间将被分配一个标签，且没有两个重叠的区间接受同样的标签。
- **定理4.6** 上述贪心算法使用与区间集合深度等量的资源为每个区间安排一个资源，这是所需资源的最优数量。

最小延迟调度：一个交换论证

再次考虑下述情况，我们有单一资源和一组使用资源的 n 个需求，每个需求需要一个时间区间. 假定资源在时刻 s 开始有效. 但是与前面的问题相反，每个需求 i 现在是更加灵活. 不是开始时间与结束时间，需求 i 有一个截止时间 d_i，并且它要求一个长为 t_i 的连续的时间区间，但是它想被安排在截止时间之前的任何时刻. 每个被接受的需求必须被分配一个长为 t_i 的时间区间，且不同的需求被分在不重叠的区间.

- 具有最小截止时间的任务优先排列：

```
按照它们的截止时间的次序对任务排序
为使符号简单，假定 d_1 ≤ ... ≤ d_n
初始，f=s
按这个次序考虑任务 i=1,2,...,n
  将任务 i 安排在从 s(i)=f 到 f(i)=f+t_i 的时间区间
  令 f=f+t_i
End
对 i=1,2,...,n 返回被安排的这组区间[s(i),f(i)]
```

- 空闲时间：机器停工了但是还有任务没有安排，称为空闲时间。
  - 存在一个没有空闲时间的最优调度。
  - 所有没有逆序也没有空闲时间的调度有相同的最大延迟。
    - 逆序：截止时间较晚的任务安排在较早截止时间的前面。
  - 存在一个既没有逆序也没有空闲时间的最优的调度。
  - 由这个贪心算法产生的调度A有最优的最大延迟。

最优超高速缓存：一个更复杂的交换论证

当处理存储分层的时候，恰好会产生这个问题. 有少量数据可以被快速存取，而大量数据需要更多的时间存取；你必须决定哪些数据块近在手边.

- 超高速缓存：在一个快速存储器中存储少量数据以便减少与一个慢速存储器的交互而花费的时间。

```
When d_i 需要被放入超高速缓存
  收回在最远的将来被需要的那个项
```

When d_i 需要被放入超高速缓存

收回在最远的将来被需要的那个项

- 最远将来规划将产生一个最优调度S。
- 简化调度：做最少的给定步所必须的工作。设S可能不是一个简化调度，令S'为S的简化，在任何一步S放入一个非必需的项d，而S'不将d移入超高速缓存。
  - S'是一个与调度S放入至多一样多个项的简化调度。
  - 对于某个数j，令S是在序列中的前j项与S_{FF}做同样收回决定的简化调度，那么存在一个在序列中的前j+1项与S_{FF}做同样收回决定的简化调度S'，并且S'不比S产生更多的缺失。
  - S_{FF}比任何其他的调度S'不产生更多的缺失，因此是最优的。

图的最短路径

最短路径问题的具体结构如下：给定有向图 $G=(V,E)$，以及一个指定的开始结点 s. 假设 s 有一条路径通向 G 中的每个其他结点. 每条边 e 有一个长度 $l_e \geqslant 0$，表示在 e 上旅行用的时间(或者距离，或者费用). 对于路径 P，P 的长度——用 $l(P)$ 表示——是 P 中所有边的长度之和. 我们的目标是确定从 s 到图中每个其他结点的最短路径. 我们应该提到，尽管问题是对有向图指定的，我们也可以处理无向图的情况，只不过将每条长为 l_e 的无向边 $e=(u,v)$ 用两条每条长度都为 l_e 的有向边 (u,v) 和 (v,u) 代替就可以了.

- Dijkstra算法：从已确认最短路径的点出发，抵达新节点的路径，

Dijkstra 算法 (G, l)

设 S 是被探查的结点的集合

对每个 $u \in S$，我们存储一个距离 $d(u)$

初始 $S=\{s\}$ 且 $d(s)=0$

While $S \neq V$

选择一个结点 $v \notin S$ 使得从 S 到 v 至少有一条边并且

$d'(v)=\min_{e=(u,v):u \in S} d(u)+l_e$ 最小

将 v 加入 S 并且定义 $d(v)=d'(v)$

Endwhile

  - 考虑在算法执行中任意一点的集合S，对每个u∈S，路径P_u是最短的s-u路径。
  - 使用优先队列，Dijkstra算法可以在具有n个节点和m条边的图上实现，它运行在O(m)时间，再加上n次ExtractMin操作和m次ChangeKey操作的时间，总时间为O(mlogn).

最小生成树问题

对于确定的对 (v_i,v_j)，我们可能以某个费用 $c(v_i,v_j)>0$ 建立 v_i 与 v_j 之间的直接链接. 于是我们可以用一个图 $G=(V,E)$ 来表示可能被建立的链接的集合. 与每条边 $e=(v_i,v_j)$ 相关的有一个正的费用 c_e. 问题就是找一个边的子集 $T \subseteq E$ 使得图 (V,T) 是连通的，且总费用 $\sum_{e \in T} c_e$ 最小(我们将假设整个图 G 是连通的；否则没有可行解).

- 令T是上述定义的网络设计问题的最小费用解，那么(V, T)是一棵树。
- Kruskal算法：初始没有边，通过费用递增添加E中的边构建生成树，如果成环则跳过边e。
  - Union-Find数据结构：给定一个节点u，操作Find(u)返回包含u的集合的名字，测试节点u和v是否在同一个集合中。操作Union(A, B)合并集合A和B.
    - MakeUnionFind(S)返回集合S上的一个Union-Find数据结构，其中所有的

元素都在分离的集合里，O(n).
        - Find(u)返回包含u的集合的名字，O(logn).
        - Union(A, B)将集合A和B合并成一个集合，O(logn).
    - 通过MakeUnionFind(S)构建最初的集合，对于每条边e=(u, v)被考虑时，计算Find(u)和Find(v)检查u和v属于不同的连通分支，之后使用Union(Find(u), Find(v))合并两个分量。
    - Kruskal算法在n个结点和m条边的图上可以在O(mlogn)运行时间内实现。
- Prim算法：从根节点s开始，每一步加入边费用最小的节点。
    - 在具有n个节点和m条边的图上，使用优先队列可以实现Prim算法，其运行时间为O(m)加上n次ExtractMin和m次ChangeKey操作的时间。总运行时间为O(mlogn).
- 逆向删除算法：从整个图(V, E)开始，从最大费用删除边，如果破坏了图的连通性，则跳过这条边。
- 割性质：假设所有的边的费用都是不等的，令S是任意节点子集，它既不是空集，也不等于全集V，令边e=(v, w)是一端在S中，另一端在V-S中的最小费用边，那么每棵最小生成树都包含边e.
- Kruskal算法产生G的一颗最小生成树，Prim算法产生一棵最小生成树，逆删除算法产生G的一颗最小生成树。
- 圈性质：假设所有边的费用都是不等的，令C是G中的一个圈，令边e=(v, w)是属于C的最贵的边，那么e不属于G的任何最小生成树。

CUE COLUMN

命题4.2的证明

证 我们将用归纳法证明这个论断. 对于 $r=1$，论断显然为真：算法由选择具有最小截止时间的需求 i_1 开始.

现在令 $r>1$. 根据归纳假设，假定这个论断对 $r-1$ 为真，我们将试图证明它对 r 也为真. 正如图 4.3 所示，由归纳假设我们有 $f(i_{r-1}) \leqslant f(j_{r-1})$. 为了使算法的第 r 个区间不更早地结束，它必须“落后”，正如该图所示. 有一个简单的理由使得它不可能发生：不选择一个结束迟的区间，贪心算法（最坏情况下）总可以选择 j_r，于是完成了归纳步骤.

我们可以更精确地如下表达这个论述. 我们知道 $f(j_{r-1}) \leqslant s(j_r)$（因为 O 由相容的区间组成）. 把它与归纳假设 $f(i_{r-1}) \leqslant f(j_{r-1})$ 组合，我们得到 $f(i_{r-1}) \leqslant s(j_r)$. 于是当贪心算法选择 i_r 时，此刻区间 j_r 是在有效区间的集合 R 中. 贪心算法选择具有最小结束时间的有效区间；因为 j_r 是这些有效区间中的一个，我们有 $f(i_r) \leqslant f(j_r)$. 这就完成了归纳步骤. ■

SUMMARIES