超能栗子
CHESTNUT EDUCATION

COMP3121

Week 1

扫码添加小助手

超能栗子
CHESTNUT EDUCATION

Contents

1.课程框架
2.分数占比
3.Week 1内容讲解

01

课程框架

基础概念：

- 时间复杂度
- 数据结构
- Searching(linear search, binary search)
- Sorting(quick, bubble, bucket…)
- Graphs(遍历方式 BFS/DFS, tree)

核心问题：

- 分治算法(Divide and Conquer)
- 贪心算法(Greedy)
- 动态规划(Dynamic Programming)
- 最大流算法(Max-flow)

超能栗子
CHESTNUT EDUCATION

02

分数占比

- Take-home quiz
  - Released week 1, attempt before your second tutorial
  - Incorporate tutor's feedback until task complete
  - Weighted 5% of course mark

- Assignments
  - Three assignments, released in weeks 1, 4 and 7
  - Each consists of 3 questions
  - Each weighted 15% of course mark

- Final Exam
  - Section I: eight MCQ
  - Section II: four algorithm design problems
  - Hurdle: must get at least 10/20 in at least one question of Section II
  - INSPERA (off campus): more info here
  - Weighted 50% of course mark

- Forum participation
  - Up to 5 bonus marks

超能栗子
CHESTNUT EDUCATION

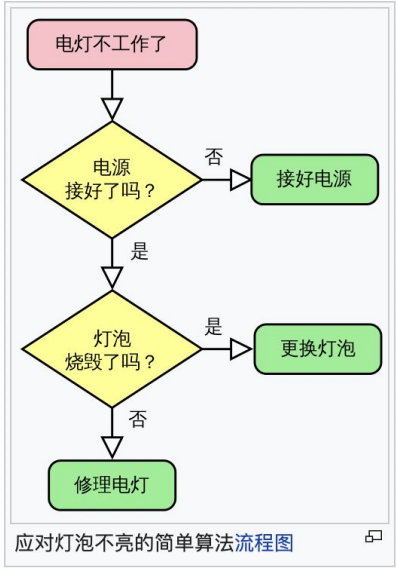
03

Week 1 讲解

超能栗子
CHESTNUT EDUCATION

What is Algorithm?

算法：解决一个问题的步骤的集合



应对灯泡不亮的简单算法流程图

小偷分赃问题

Problem:

Two thieves have robbed a warehouse and have to split a pile of items without price tags on them. Design an algorithm for doing this in a way that ensures that each thief **believes** that he has got at least one half of the loot.

两个小偷中的一个把这堆东西分成他认为价值相等的两部分
,
然后另一个小偷选择他认为更好的那部分。

小偷分赃问题

Problem:

Three thieves have robbed a warehouse and have to split a pile of items without price tags on them. How do they do this in a way that ensures that each thief believes that he has got at least one third of the loot?

三个小偷呢？

Algorithm:

T_1 makes a pile P_1 which he believes is $1/3$ of the whole loot;
T_1 proceeds to ask T_2 if T_2 agrees that $P_1 \leq 1/3$;
If T_2 says YES, **then** T_1 asks T_3 if T_3 agrees that $P_1 \leq 1/3$;
    If T_3 says YES, **then** T_1 takes P_1;
        T_2 and T_3 split the rest as in Problem 1.
    Else if T_3 says NO, **then** T_3 takes P_1;
        T_1 and T_2 split the rest as in Problem 1.
Else if T_2 says NO, **then** T_2 reduces the size of P_1 to $P_2 < P_1$ such that T_2 thinks $P_2 = 1/3$;
    T_2 then proceeds to ask T_3 if he agrees that $P_2 \leq 1/3$;
      If T_3 says YES **then** T_2 takes P_2;
        T_1 and T_3 split the rest as in Problem 1.
      Else if T_3 says NO then T_3 takes P_2;
        T_1 and T_2 split the rest as in Problem 1.

算法中证明的作用

- However, sometimes it is **NOT** clear from a description of an algorithm that such an algorithm will not enter an infinite loop and fail to terminate.
- Sometimes it is **NOT** clear that an algorithm will not run in exponentially many steps (in the size of the input), which is usually almost as bad as never terminating.
- Sometimes it is **NOT** clear from a description of an algorithm why such an algorithm, after it terminates, produces a desired solution.
- Proofs are needed for such circumstances; in a lot of cases they are **the only way** to know that the algorithm does the job.
- For that reason we will **NEVER** prove the obvious (the CLRS textbook sometimes does just that, by sometimes formulating and proving trivial little lemmas, being too pedantic!). We will prove only what is genuinely nontrivial.
- **However, BE VERY CAREFUL what you call trivial!!**

1. 不确定这个算法会不会进入无限循环，无法终止
2. 不确定这个算法的时间复杂度是不是指数级别的
3. 不确定这个算法是如何产生我们想要的结果的

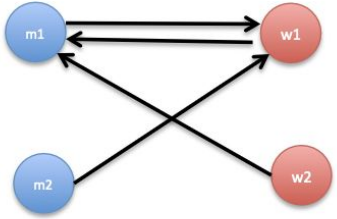
Stable Matching Problem: Gale-Shapley algorithm

- Assume that you are running a dating agency and have n men and n women as customers;
- They all attend a dinner party; after the party:
  - every man gives you a list with his ranking of all women present, **and**
  - every woman gives you a list with her ranking of all men present;
- Design an algorithm which produces a *stable matching*, which is: a set of n pairs $p = (m, w)$ of a man m and a woman w so that the following situation never happens:

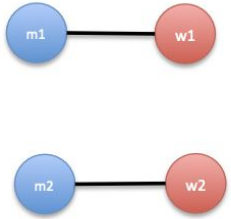
  for two pairs $p = (m, w)$ and $p' = (m', w')$:
  - man m prefers woman w' to woman w, **and**
  - woman w' prefers man m to man m'.

Case1:

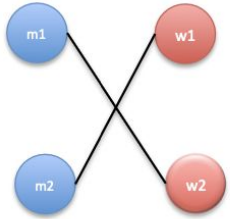
Preferences:



stable matching

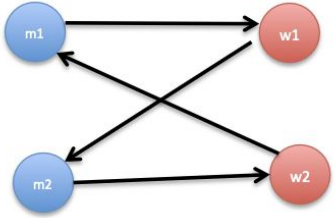


not stable

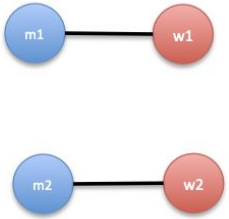


Case2:

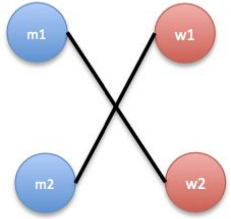
Preferences:



stable matching



also stable!



超能栗子
CHESTNUT EDUCATION

暴力法

现在有 n 个男人和 n 个女人，如果把他们两两配对，一共有 n!种情况。考察每一种情况，如果找到一种每一组都是稳定配对的情况，就用这种情况配对他们。

超能栗子
CHESTNUT EDUCATION

Gale-Shapley algorithm

假设男人们依次，由从各自心中魅力 值从高到低的顺序，向女人们发出约会邀请。

当一个女人没有 约会对象时，不论哪个男人向她 发出邀请她都会接受，毕竟有总比没有好。

当一个男人还没有约会对象时，我们把他称作“*自由人*” (free man)。

当一个女人已 经有了约会对象，却被一个在她心中魅力 值更高的男人邀 请时，她将选择更有魅力的男人作 为约会对象。原先的约会对象变回”*自由人*“。

每个人只能有一个 约会对象。当不再有自由人 时，匹配结束。

Produces pairs in stages, with possible revisions;

- A man who has not been paired with a woman will be called *free*.
- Men will be proposing to women.Women will decide if they accept a proposal or not.
- Start with all men free;

While there exists a free man who has not proposed to all women
pick such a free man m and have him propose to the highest ranking woman w on his list to whom he has not proposed yet;
If no one has proposed to w yet
she always accepts and a pair $p = (m, w)$ is formed;
Else she is already in a pair $p' = (m', w)$;
If m is higher on her preference list than m'
the pair $p' = (m', w)$ is deleted;
m' becomes a free man;
a new pair $p = (m, w)$ is formed;
Else m is lower on her preference list than m';
the proposal is rejected and m remains free.

算法中存在的问题的证明

Claim 1: Algorithm terminates after $\leq n^2$ rounds of the *While* loop
Proof:

- In every round of the *While* loop one man proposes to one woman;
- every man can propose to a woman at most once;
- thus, every man can make at most n proposals;
- there are n men, so in total they can make $\leq n^2$ proposals.

Thus the *While* loop can be executed no more than n^2 many times.

Claim 2: Algorithm produces a matching, i.e., every man is eventually paired with a woman (and thus also every woman is paired to a man)
Proof:

- Assume that the while *While* loop has terminated, but m is still free.
- This means that m has already proposed to every woman.
- Thus, every woman is paired with a man, because a woman is not paired with anyone only if no one has made a proposal to her.
- But this would mean that n women are paired with all of n men so m cannot be free. **Contradiction!**

算法中存在的问题的证明

Claim 3: The matching produced by the algorithm is stable.
Proof: Note that during the *While* loop:

- a woman is paired with men of increasing ranks on her list;
- a man is paired with women of decreasing ranks on his list.

Assume now the opposite, that the matching is not stable;
thus, there are two pairs $p = (m, w)$ and $p' = (m', w')$ such that:

$$m \text{ prefers } w' \text{ over } w;$$
$$w' \text{ prefers } m \text{ over } m'.$$

- Since m prefers w' over w, he must have proposed to w' before proposing to w;
- Since he is paired with w, woman w' must have either:
  - rejected him because she was already with someone whom she prefers, or
  - dropped him later after a proposal from someone whom she prefers;
- In both cases she would now be with m' whom she prefers over m.
- **Contradiction!**

Asymptotic notation (渐进符号)

- **"Big Oh" notation:** $f(n) = O(g(n))$ is an abbreviation for:

  "There exist positive constants c and n_0 such that $0 \leq f(n) \leq c\, g(n)$ for all $n \geq n_0$".

- In this case we say that $g(n)$ *is an asymptotic upper bound for* $f(n)$.

- $f(n) = O(g(n))$ means that $f(n)$ does not grow substantially faster than $g(n)$ because a multiple of $g(n)$ eventually dominates $f(n)$.

- Clearly, multiplying constants c of interest will be larger than 1, thus "enlarging" $g(n)$.

Asymptotic notation (渐进符号)

- **"Omega" notation:** $f(n) = \Omega(g(n))$ is an abbreviation for:

  "There exists positive constants c and n_0 such that $0 \leq c\, g(n) \leq f(n)$ for all $n \geq n_0$."

- In this case we say that $g(n)$ *is an asymptotic lower bound for* $f(n)$.

- $f(n) = \Omega(g(n))$ essentially says that $f(n)$ grows at least as fast as $g(n)$, because $f(n)$ eventually dominates a multiple of $g(n)$.

- Since $c\, g(n) \leq f(n)$ if and only if $g(n) \leq \frac{1}{c} f(n)$, we have $f(n) = \Omega(g(n))$ if and only if $g(n) = O(f(n))$.

- **"Theta" notation:** $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$; thus, $f(n)$ and $g(n)$ have the same asymptotic growth rate.

超能栗子
CHESTNUT EDUCATION

Thank You
For Watching
感/谢/您/的/观/看

Speaker: 超能栗子