Greedy Algorithm
貪婪演算法

林劭原老師

林劭原 老師

前言

- Three useful algorithm design techniques:
- divide-and-conquer approach
- dynamic programming
- greedy method

Greedy Algorithm

- Always makes the choice that looks best at the moment
- makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution.
- 「貪婪演算法」永遠選擇目前看起來最好的那個選擇。它選擇的是a locally optimal choice，並希望這個 locally optimal choice 能得到 a globally optimal solution。

Greedy Algorithm

- Dynamic programming considers ”all possible ways” to derive an optimal solution.
- It uses a tables(s) to avoid re-computing solutions of sub-problems.

Greedy Algorithm

- 小刀：greedy strategy, 大刀：dynamic programming
- For many optimization(優化) problems, the greedy strategy is sufficient to derive an optimal solution and it is not necessary to use dynamic programming.
- 兩種背包問題
- 0-1 knapsack problem:無法用greedy strategy解決，必須dynamic programming
- Fractional knapsack problem:greedy strategy 就能解決

The fractional knapsack problem(可以只取一部份)

- Greedy strategy for fractional knapsack problem
- Among un-chosen items, chose the one with the highest v_i/w_i.
- If $w_i \leq$ current capacity of the knapsack, then put all of item i into the knapsack; otherwise, put only the amount of current capacity of item i into the knapsack.
- *Among un-chosen items, chose the one with the highest v_i (does not work)
- *Among un-chosen items, chose the one with the smallest w_i (does not work)

The fractional knapsack problem(可以只取一部份)

- 例:capacity W = 50 pounds.
- item weight w_i value v_i v_i/w_i

item	weight w_i	value v_i	v_i/w_i
1	10 pounds	\$60	6
2	20 pounds	\$100	5
1	30 pounds	\$120	4

- Optimal solution:60+100+120*2/3

The activity-selection problem (活動挑選問題)

- Let $S = \{a_1, a_2, \ldots, a_n\}$ be n activities that use a resource.這些活動共用某資源
- Activity a_i has a start time s_i and a finish time f_i, where $0 \leq s_i < f_i < \infty$
- Activity a_i and a_j are compatible(可匹配的) if $s_i \geq f_j$ or $s_j \geq f_i$
- The activity-selection problem is to select a maximum-size subset of mutually compatible activities.最多個可互相匹配的工作
- 例如: $S = \{a_1, a_2, a_3, a_4\}$

i	1	2	3	4
s_i	1	3	0	5
f_i	4	5	6	7

- The optimal solution is $\{a_1, a_4\}$ or $\{a_2, a_4\}$.

Greedy strategy for activity-selection problem

- Among un-chosen activities, chose the one with the smallest f_i and compatible to all chosen activities. --- works
- Among un-chosen activities, chose the one with the largest s_i and compatible to all chosen activities. --- works
- Among un-chosen activities, chose the one with the least duration and compatible to all chosen activities. --- does not work
- Among un-chosen activities, chose the one with the least overlap and compatible to all chosen activities. --- does not work

Greedy strategy for activity-selection problem

- Assume that activities in S are ordered such that $f_1 \leq f_2 \leq \cdots \leq f_n$; otherwise sort these activities so that $f_1 \leq f_2 \leq \cdots \leq f_n$.
- Let A be the solution.
- n = S.length
A = $\{a_1\}$ // This greedy strategy will always choose a_1
k = 1
for m = 2 to n do
    if $s_m \geq f_k$ //a_m is compatible to all the chosen activities
        A = AU $\{a_m\}$
        k = m;
return A

Greedy strategy for activity-selection problem

- Proof of correctness 很重要!
- To prove the correctness of a greedy algorithm, one can prove that any optimal solution can be modified into a greedy solution without losing the optimality. 證明某個貪婪演算法能得到最佳解的常用方法是:證明任何的最佳解，均可被修改成貪婪演算法的解，不失其最佳性。若是如此，貪婪演算法的解得到的就是最佳解。

- Theorem. This greedy algorithm produces an optimal solution.
- Proof. Note that we assume $f_1 \leq f_2 \leq \cdots \leq f_n$.
  Consider an arbitrary optimal solution B.
  Order the activities in B by their finish times.
  Suppose the first activity in B is a_k.
  If $\mathrm{k} = 1$, then B begins with a greedy choice, i.e., with a_1.
  If $\mathrm{k} \neq 1$, then let $B' = B - \{a_k\} \cup \{a_1\}$.
  Since $f_1 \leq f_k$, all the activities in B' are compatible.
  Since $|B'| = |B|$, B' is an optimal solution that begins with a greedy choice, i.e. , with a_1.
  Thus, there exists an optimal solution that begins with a greedy choice, i.e., with a_1.
  Once we choose a_1, the problem reduces to finding an optimal solution of $S' = \{a_i \in$