Algorithm and Object-Oriented Programming for Modeling

Part 6: Greedy Algorithm

MSDM 5051, Yi Wang (王一), HKUST


我全都要

Greedy Algorithm

活在当下

Design strategy such that

local best choice == global best choice

How to pay 245 with minimal # bills

Say, we have 100, 20 and 5 bills.

How to pay 245 with minimal # bills

Say, we have 100, 20 and 5 bills.

Assume solution

$m \times 100 + n \times 20 + k \times 5 = 245$,

Then

$(m + 1) \times 100 + (n - 5) \times 20 + k \times 5 = 245$,

So the more 100 bills, the better (as long as $n > 0$)

Example: activity arrangement

Each activity i has a start time s_i and an end time f_i.

How to arrange the largest number of non-overlapping activities?

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16

Variants:

How to remove intervals to make the remaining nonoverlapping

Example: activity arrangement

Each activity i has a start time s_i and an end time f_i.

How to arrange the largest number of non-overlapping activities?

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16

Idea:

(1) Arrange one with earliest ending time

(2) In the remaining non-overlapping activities, do the same

Example: activity arrangement

Each activity i has a start time s_i and an end time f_i.

How to arrange the largest number of non-overlapping activities?

Idea:

(1) Arrange one with earliest ending time

(2) In the remaining non-overlapping activities, do the same

Why it's correct?

Example: activity arrangement

Each activity i has a start time s_i and an end time f_i.

How to arrange the largest number of non-overlapping activities?

Idea:

(1) Arrange one with earliest ending time

(2) In the remaining non-overlapping activities, do the same

Why it's correct?

Assume $i \to j \to$ ... is a solution. Then for k with an earlier end time $f_k < f_i$, $k \to j \to$ ... is also a solution.

General procedure of a proof:

Aim: ∃ solution **starting** with a greedy choice

May have multiple solutions. Just find one

Let recursion do the rest

Steps: (1) Assume a solution S

(2) Prove that S’ is also a solution,

where S’ starts with a greedy choice

Assume $i \rightarrow j \rightarrow$ ... is a solution. Then for k with an earlier end time $f_k < f_i$, $k \rightarrow j \rightarrow$ ... is also a solution.

45. Jump Game II

Medium 6154 233 Add to List Share

Given an array of non-negative integers `nums`, you are initially positioned at the first index of the array.

Each element in the array represents your maximum jump length at that position.

Your goal is to reach the last index in the minimum number of jumps.

You can assume that you can always reach the last index.

Example 1:

```
Input: nums = [2,3,1,1,4]
Output: 2
Explanation: The minimum number of jumps to reach the last index is 2. Jump 1 step from index 0 to 1, then 3 steps to the last index.
```

Example 2:

```
Input: nums = [2, 3, 0, 9, 4]
Output: 2
```

Solution:

Starting from the last position. Each time jump back farthest.

Solution:

Starting from the last position. Each time jump back farthest.

Why?

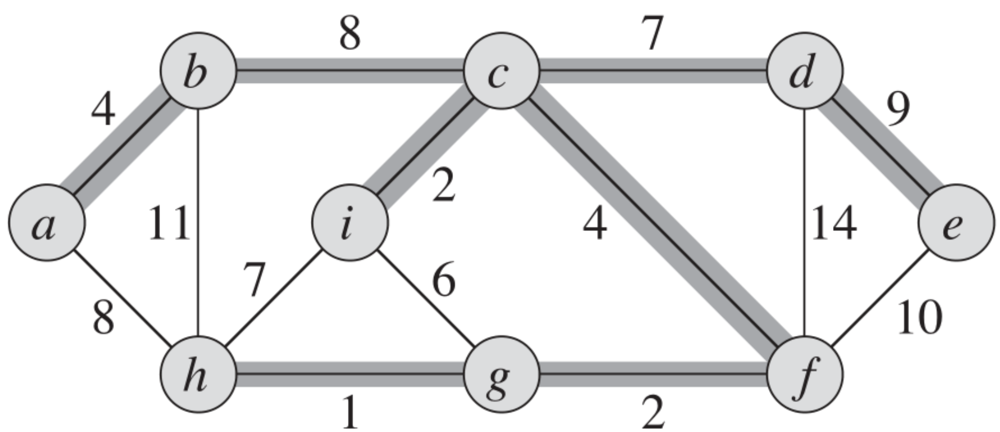
Assume $j_1 \rightarrow j_2 \rightarrow \ \ldots \rightarrow j_{n-1} \rightarrow j_n$ is the solution.

Then replacing j_n with the farthest jump back (denoted by k_n)

is also a solution, because k_n is also reachable by j_{n-1}.

Recall some other greedy algorithms

Minimum Spanning Tree

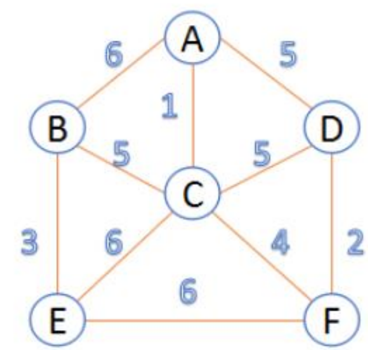


For weighted & undirected diagram,

find a subset of edges to connect all vertices with minimal total weight

Kruskal: add smallest edges & merge trees

```
MST-KRUSKAL(G, w)          # G: graph, w: weight
A = Ø   # The set of edges that finally makes the MST
for each vertex v ∈ G.V
    MAKE-SET(v)            # make a tree for each vertex
sort the edges of G.E into nondecreasing order by weight w    # greedy
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)     # if same tree, will form loop
        A = A ∪ {(u, v)}
        UNION(u, v)        # merge two trees
return A
```



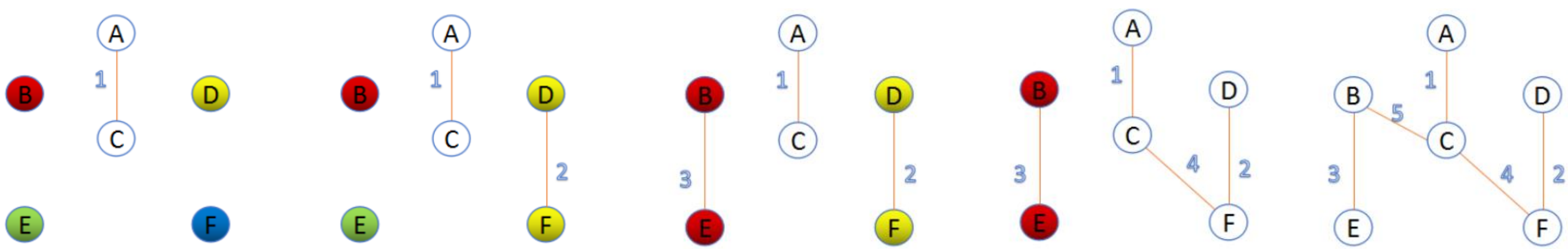
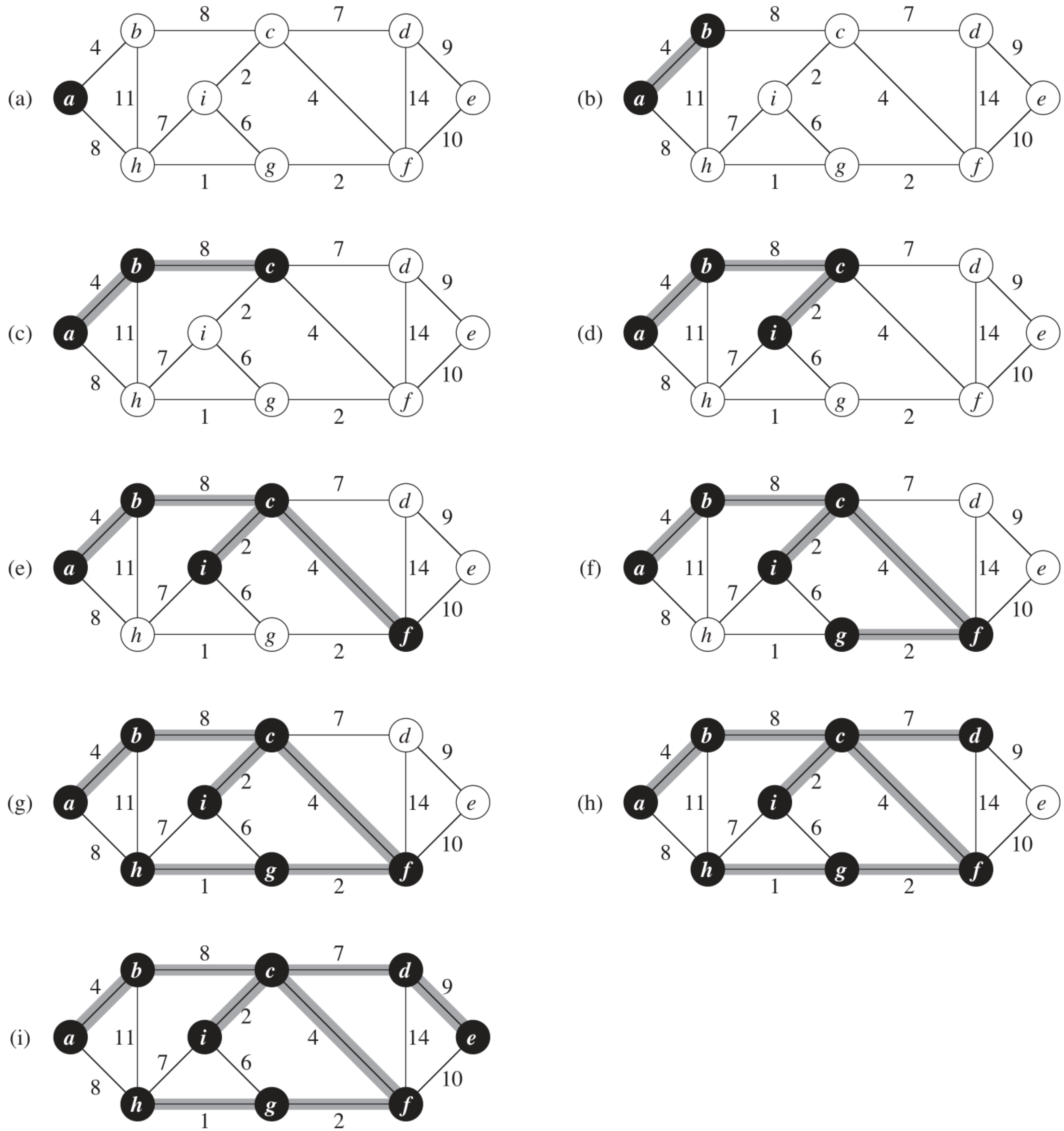


Image credit: https://blog.csdn.net/luoshixian099/article/details/51908175

Prim: add vertex with minimal distance to *one tree*, until tree spanning graph

```
MST-PRIM(G, w, r)            # r: any given root vertex
for each u ∈ G.V
    u.key = ∞                # key: minimal distance to the existing tree
    u.π = NIL                # π: parent of u in the tree
r.key = 0
Q = G.V                      # Q: vertices to be added, min-priority queue
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key          # update distances to tree (also update Q)
            v.π = u
            v.key = w(u, v)
```


(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)

Greedy as approximate solutions: knapsack

Summary of Greedy Algorithm:

(1) Need greedy strategy

(2) Easy to write

(3) Proof: exist a solution starts from greedy

(4) Approximate solutions