计算机问题求解 — 论题2-1
- 算法方法

2014年2月18日

方法与技术(结构)

- 问题：
  - 给定一群人(例如：在一个大操场上)，给定一个数值(例如：175)，输出高度恰好等于该数值的人。
- 方法：
  - 搜索
- 但是我们仍然需要明确，用什么样的方式来实现“搜索”

问题1：

你能解释下面的话吗？

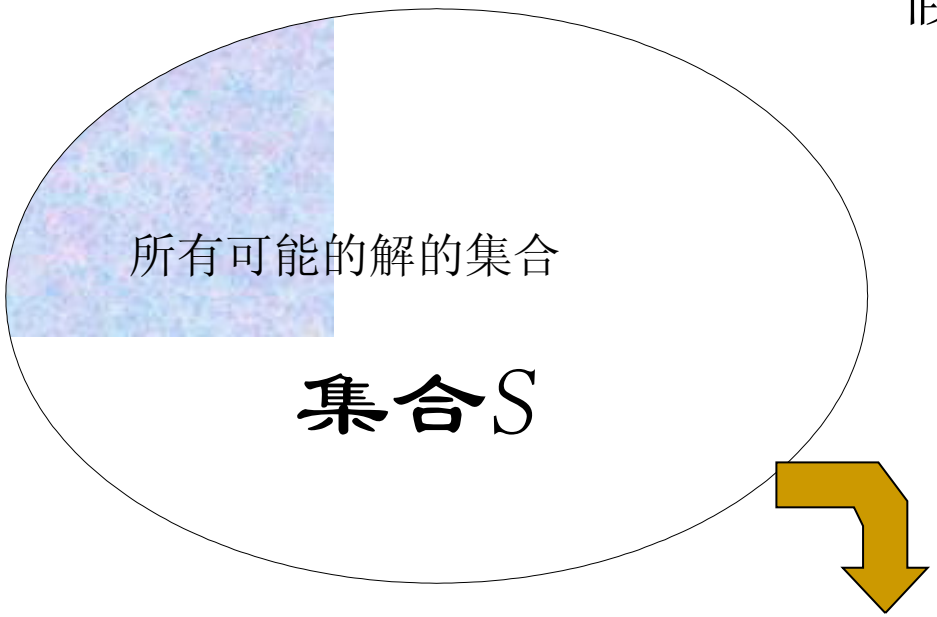
It is all very well talking about the *constructs* that an algorithm may use—that is, the pieces it might be composed of—but we must say something more about the ways of going about using these pieces to make a whole.

搜索“解空间” – 一个例子

- 一位父亲请一位数学家猜他3个孩子的年龄，他提示说：
  - 3人年龄的乘积是36。
  - 这时他们恰好经过一幢房子，父亲又提示说：他们年龄之和等于这房子窗户的个数13。
- 父亲见数学家仍然犹豫，又补充说：
  - 老大很小的时候家中没有其他孩子跟他一起玩。
- 你能说出3个孩子的年龄吗？

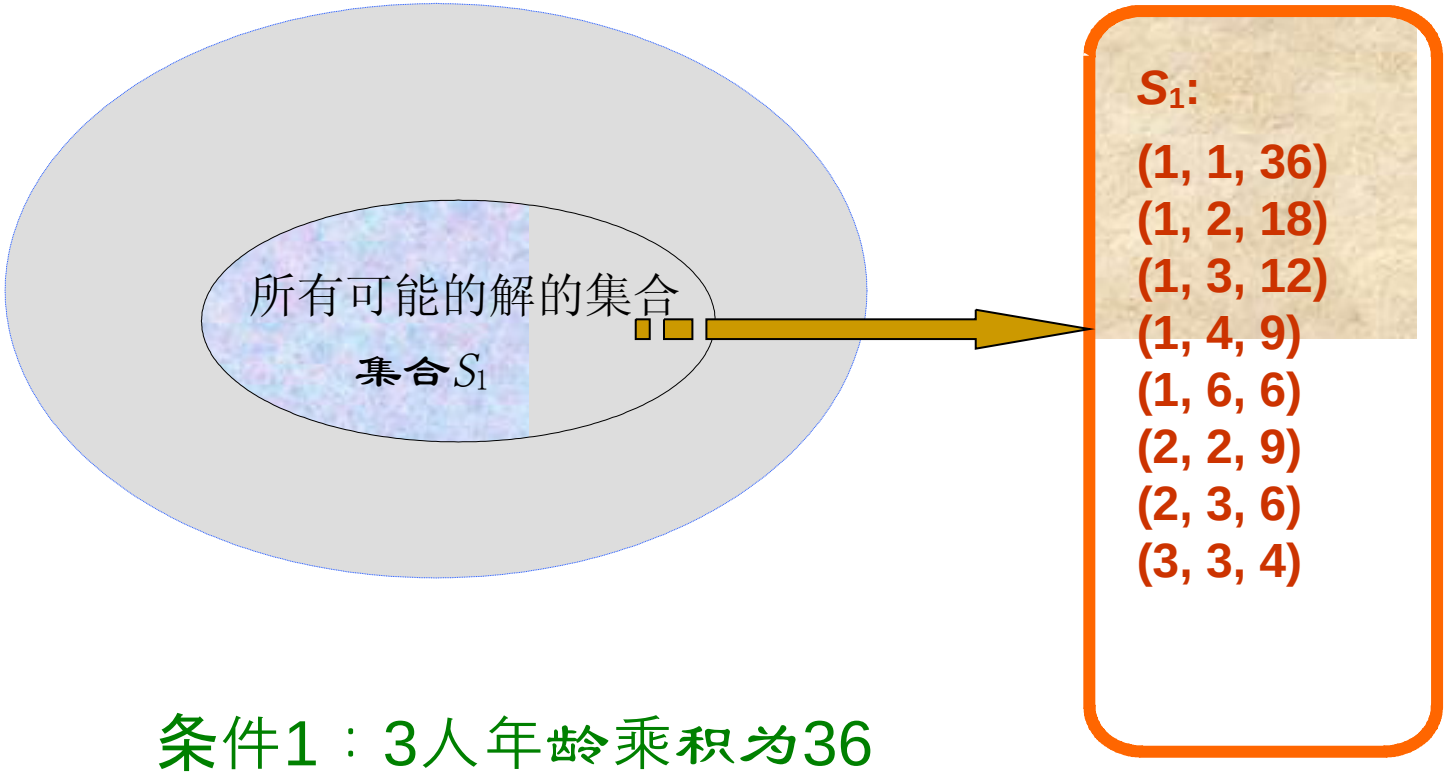
初始的解空间

假设年龄精确到整数



$S = \{ (i, j, k) \mid i, j, k$ 是非负整数$\}$

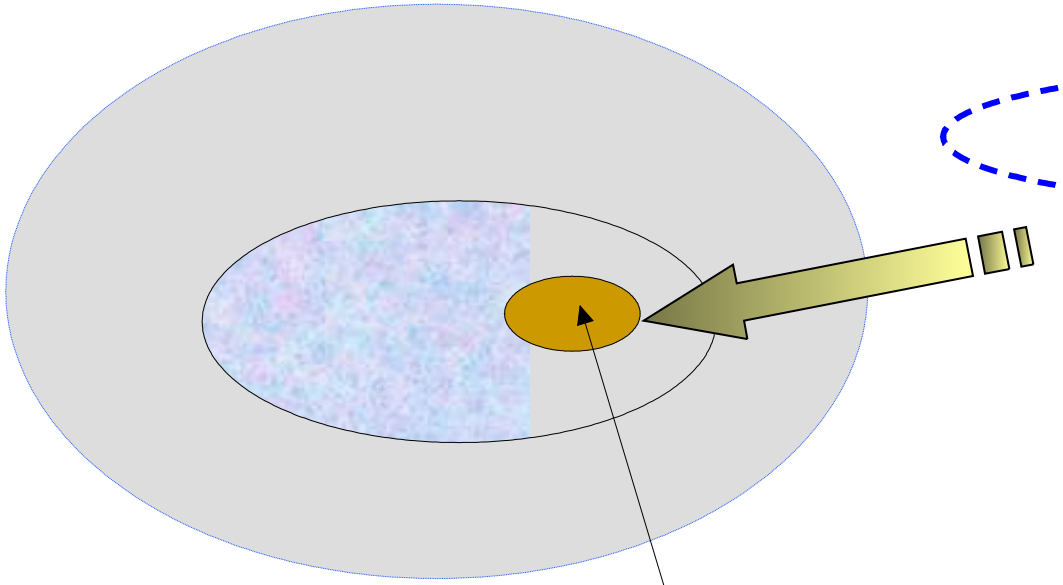
利用条件缩小可能的解空间



条件1：3人年龄乘积为36

解空间还有缩小的可能

尽管已经知道了年龄之和, 那个数学家仍然说不出答案...



可能的解的集合

S_1:

(1, 1, 36)	38
(1, 2, 18)	21
(1, 3, 12)	16
(1, 4, 9)	14
(1, 6, 6)	13
(2, 2, 9)	13
(2, 3, 6)	11
(3, 3, 4)	10

再进一步就是解！

- 当前可能的解的集合：

  { (1,6,6), (2,2,9) }

- 但是：老大没有同年龄的兄弟姐妹.

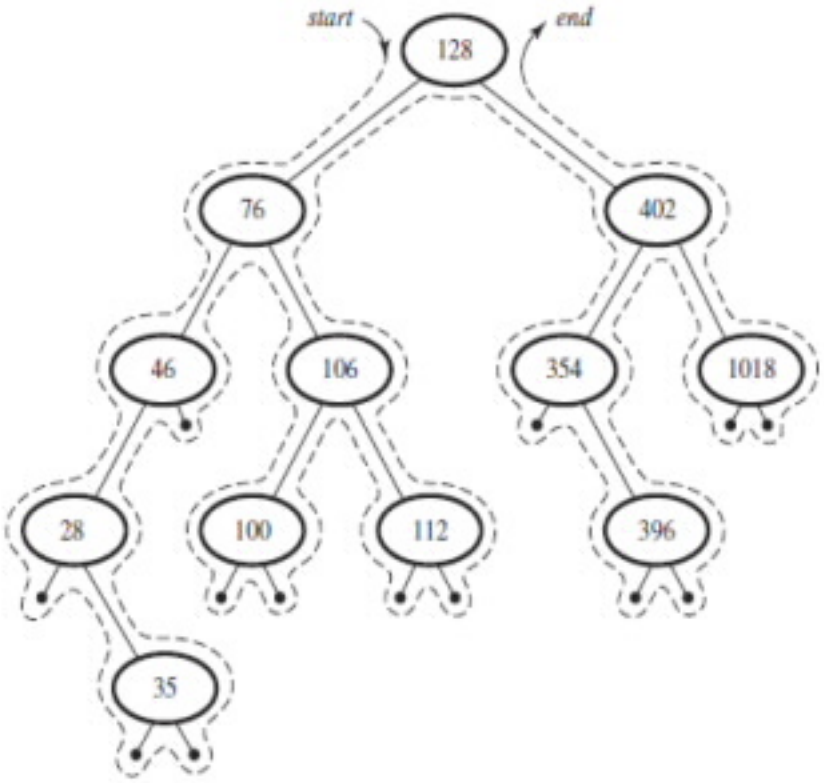
- 因此三个孩子的年龄分别是：

  9 岁、 2 岁和 2 岁

问题求解的基本“方法”

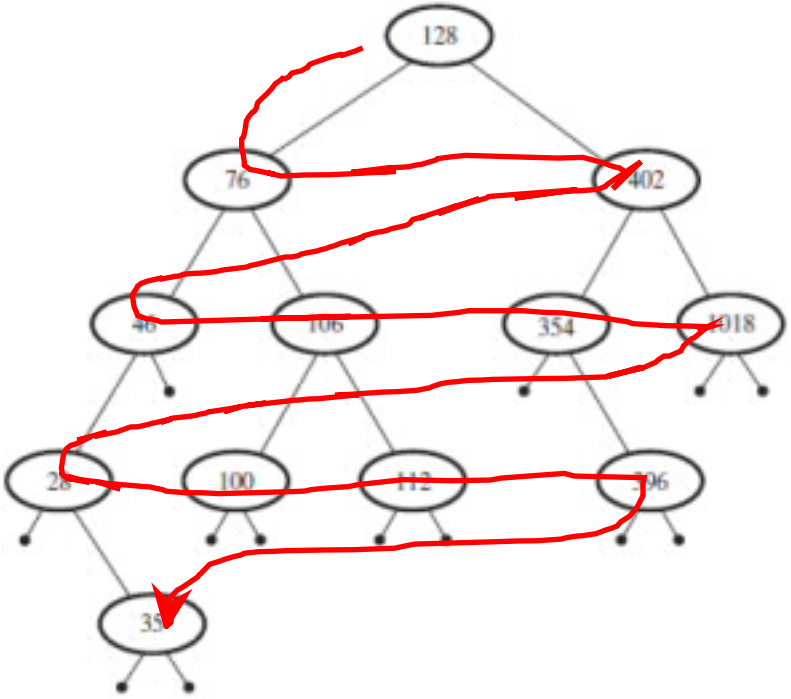
- 确定合理的解空间，并表示为某种“结构”。
- 利用已知的限制条件（知识）尽可能快的压缩可能的解空间。
  - 当解空间已经足够小，我们就可以“直接”解题。
- 如果很难确定解空间的范围，或者很难有效地缩小解空间，这个题目就“很难”。

搜索结构



深度优先 - DFS

广度优先 - BFS

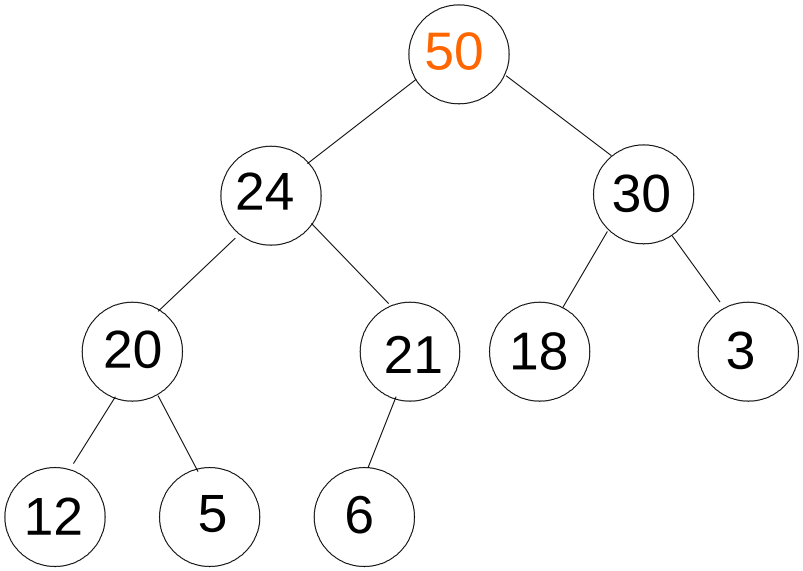


“聪明”的搜索结构

二分搜索树 - BST

堆　Heap

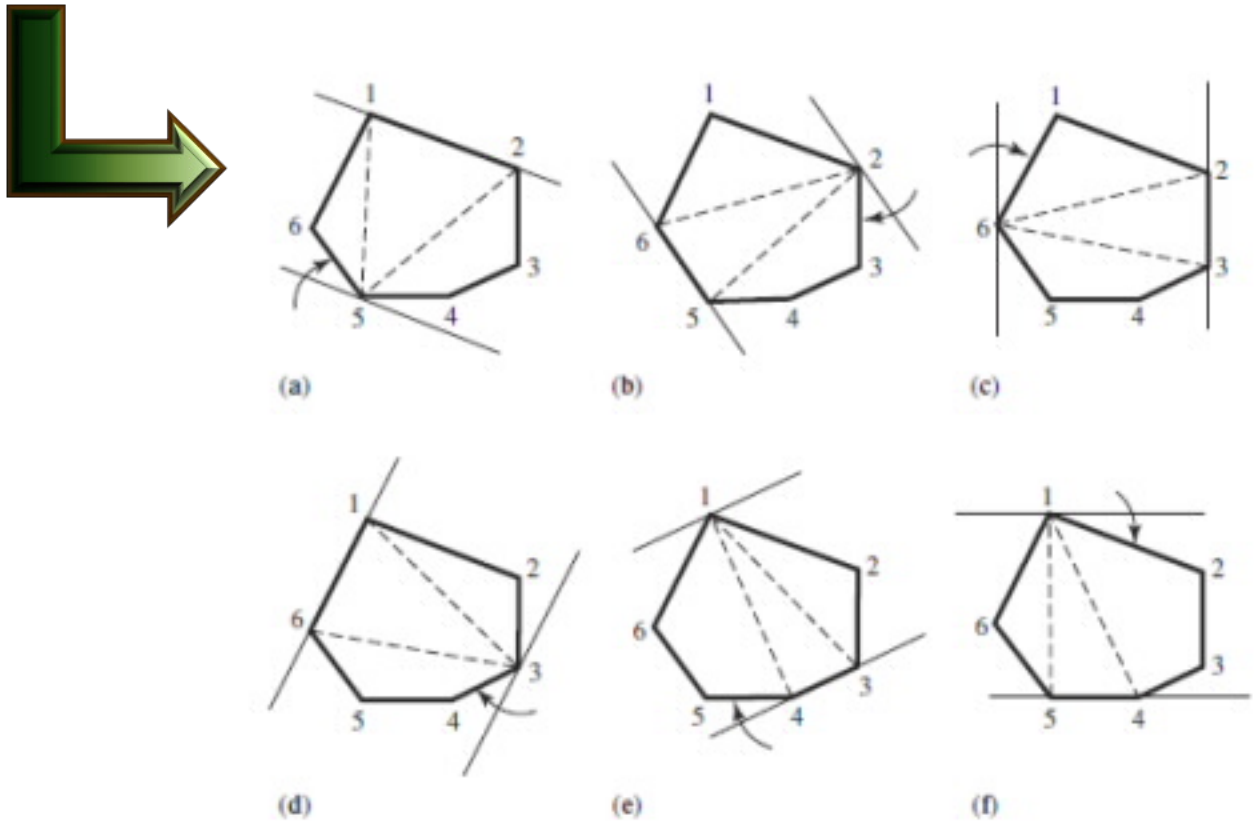
优先队列的一种实现



问题2：

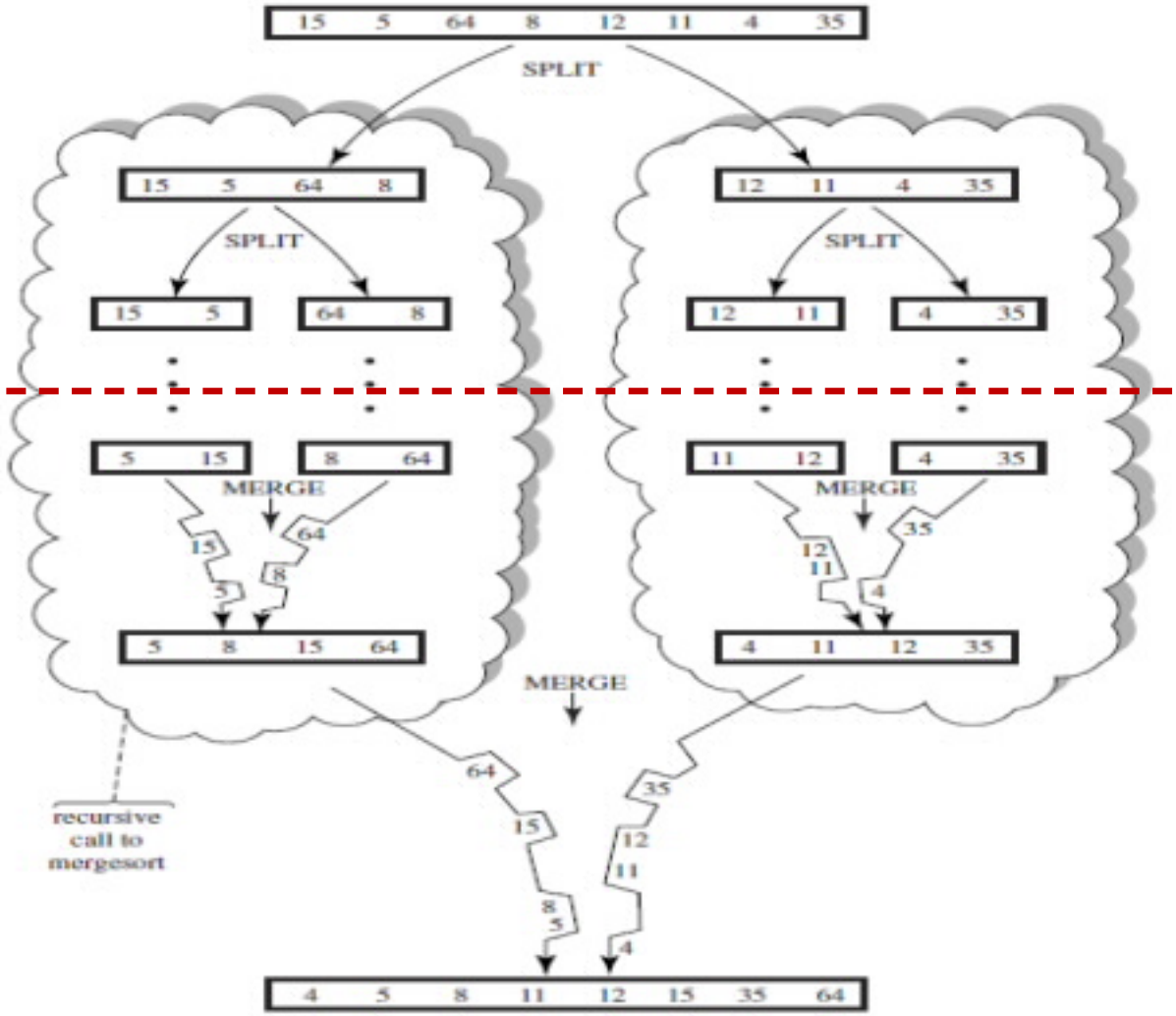
你能解释一下解Maximal Polygon Distance问题的过程中是如何建立并缩小解空间的吗？

As the maximum distance will clearly occur for two of the vertices (why?), there is no need to look at any points along the polygon's edges other than the vertices.

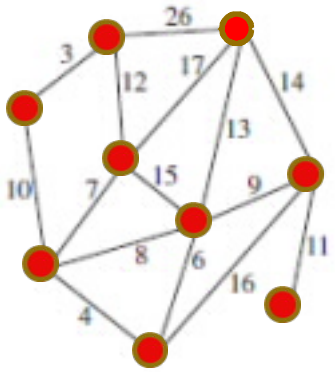


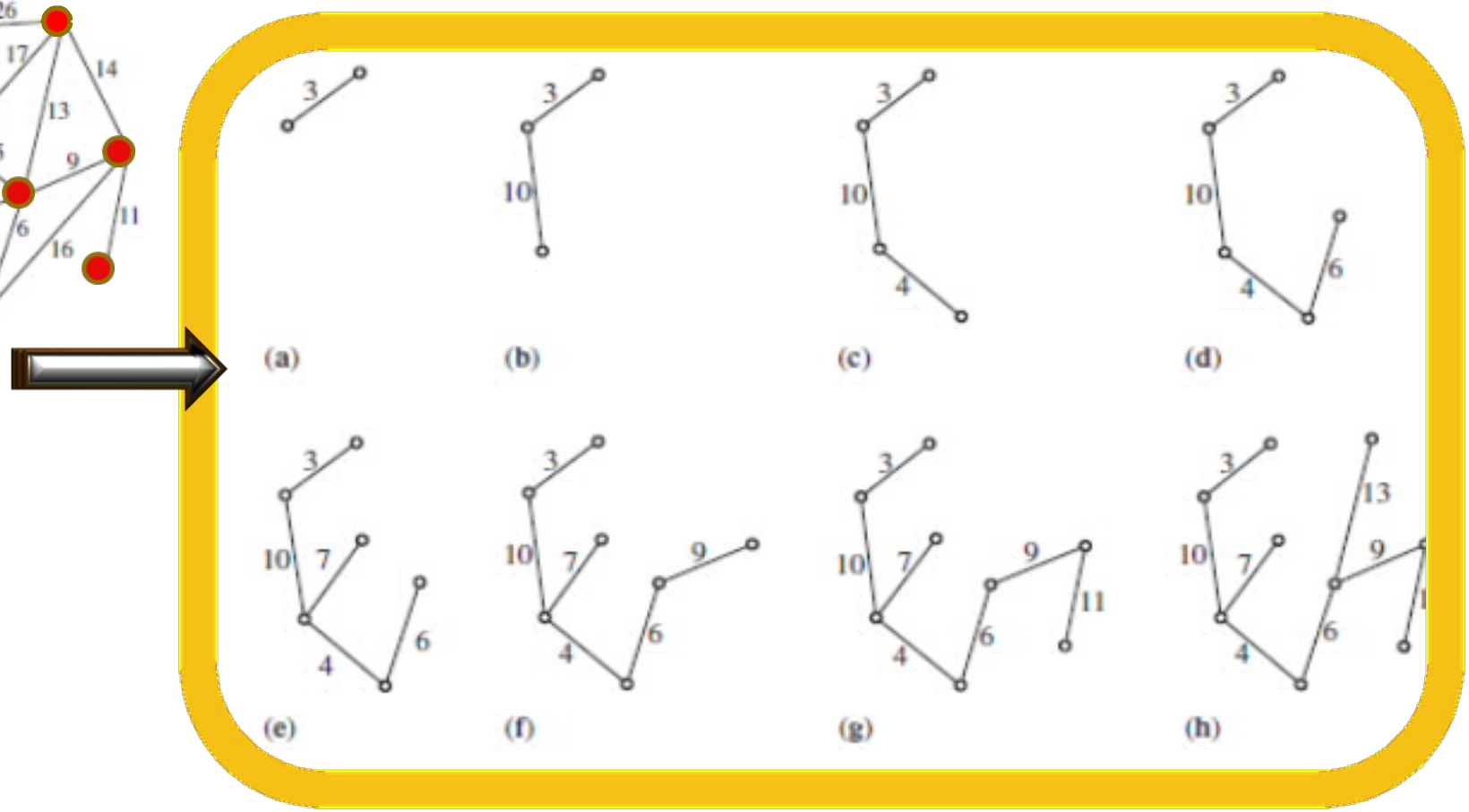
This example was chosen to further illustrate that recognizing the need for a traversal, and figuring out what really is to be traversed, is important and can be of considerable help, but it does not always suffice when it comes to solving tricky algorithmic problems; some insight and a good deal of knowledge of the relevant subject matter can do no harm.

Mergesort: Divide-and-Conquer

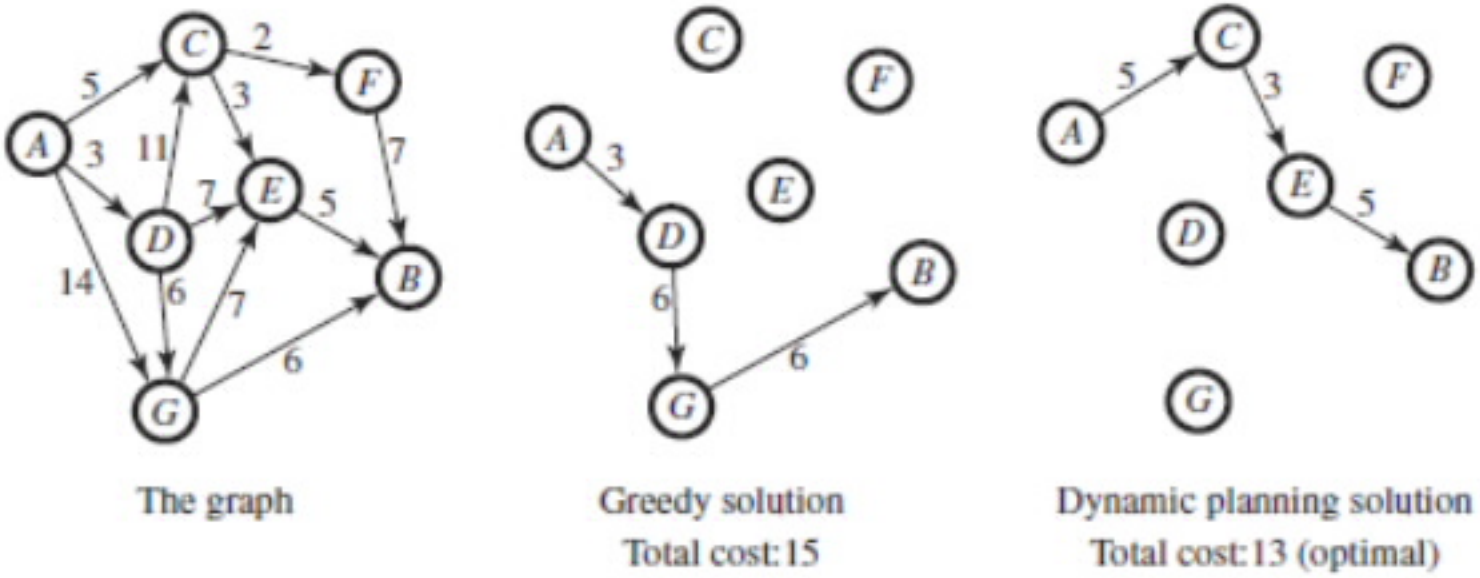


Greedy: Minimal Spanning Tree





Greedy: Simple, but may Fail!



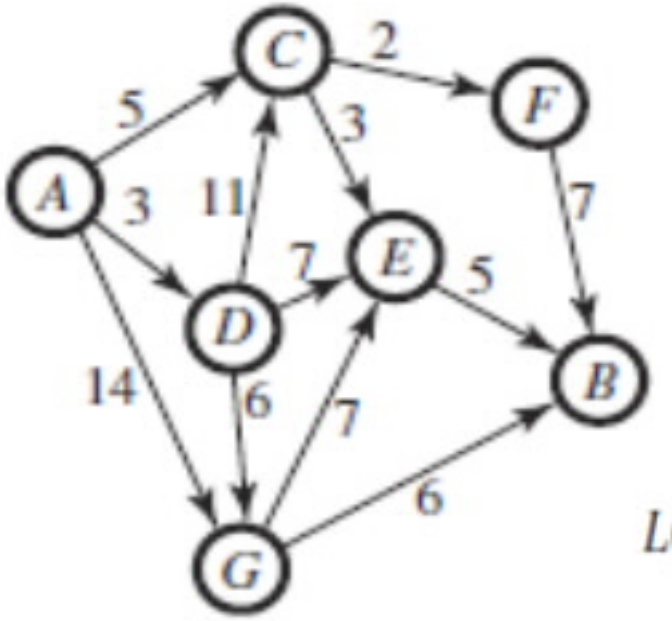
The graph

Greedy solution
Total cost: 15

Dynamic planning solution
Total cost: 13 (optimal)

问题3：
你能从“搜索”的角度说明为什么Greedy可能Fail吗？

动态规划



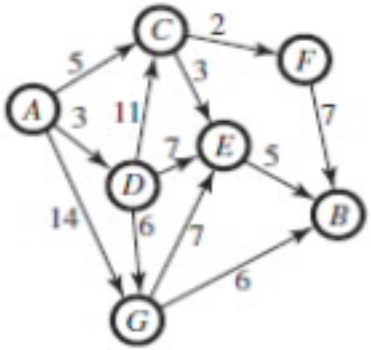
The graph

$L(A) = \text{minimum of: } 5 + L(C), 14 + L(G), 3 + L(D)$

$L(D) = \text{minimum of: } 7 + L(E), 6 + L(G), 11 + L(C)$

问题4：

用 Dynamic Programming解最短通路问题为什么就不会出错了？



The graph

$$L(A) = \text{minimum of: } 5 + L(C), 14 + L(G), 3 + L(D)$$

$$L(D) = \text{minimum of: } 7 + L(E), 6 + L(G), 11 + L(C)$$

问题5：

既然Dynamic Programming本质上是 exhaustive，为什么还能保证效率可以接受？

问题6：

你阅读的材料中介绍了其它几个“算法方法”。你能从“搜索”的角度对它们加以解释吗？

Divide-and-Conquer; Greedy; Dynamic Programming; Using “clever” data structure

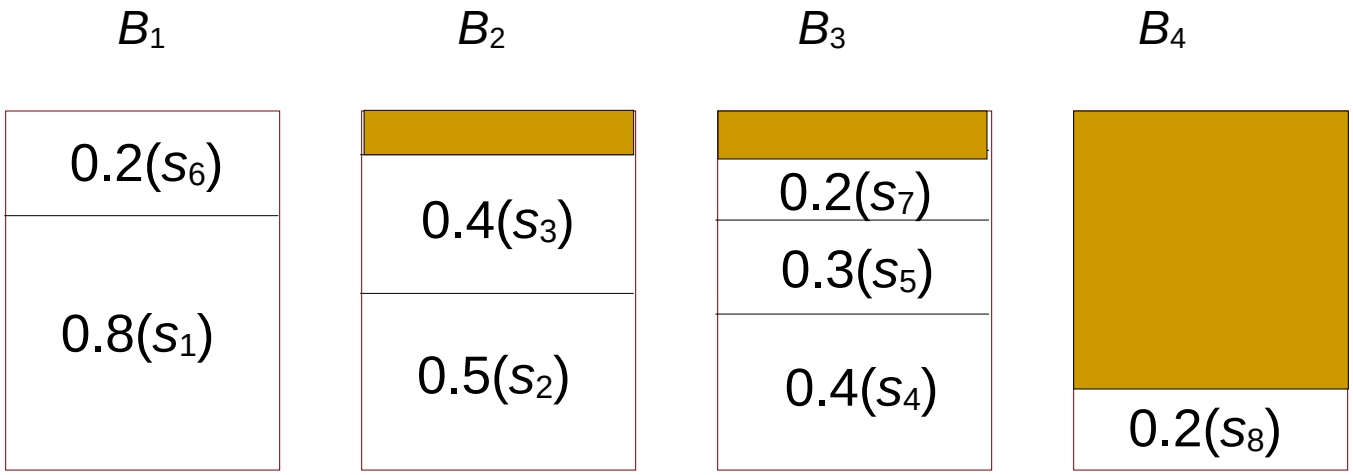
用Greedy解“难”题

- Bin Packing Problem
  - Suppose we have an unlimited number of bins each of capacity one, and *n* objects with sizes s_1, s_2, …, s_n where $0<s_i$ 1 (s_i are rational numbers)
  - ***Optimization problem***: Determine the smallest number of bins into which the objects can be packets (and find an optimal packing) .
  - Bin packing is a NPC problem

问题7：为什么这是难题？

First Fit Decreasing - FFD

- The strategy: packing the largest as possible
- Example: S=(0.8, 0.5, 0.4, 0.4, 0.3, 0.2, 0.2, 0.2)



This is ***NOT*** an optimal solution!

但可以证明：也不是太差！

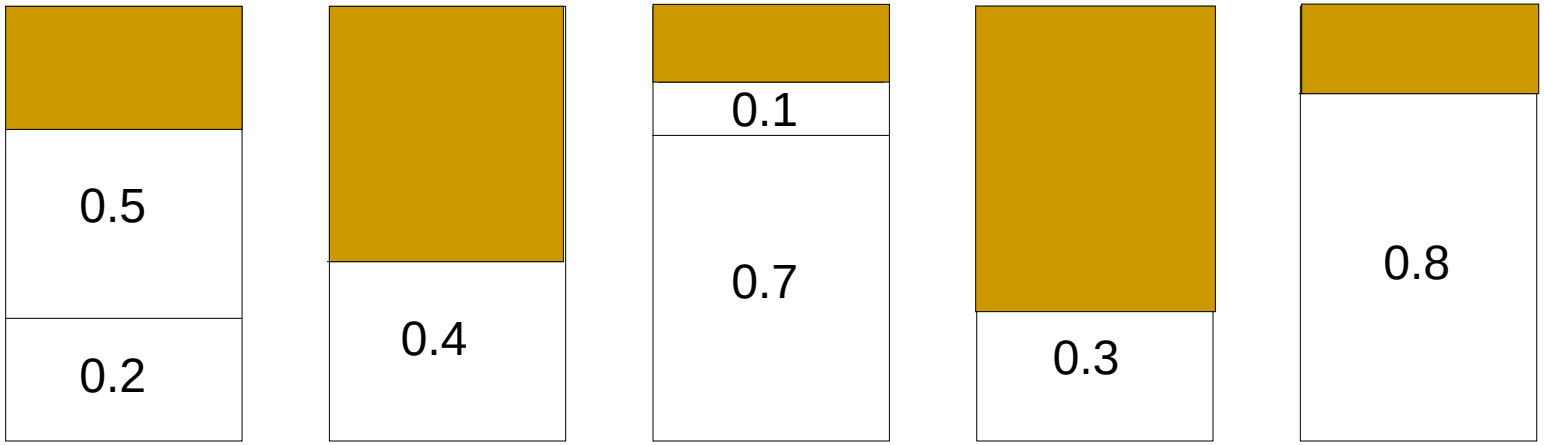
Online: 会更困难

问题8：

你是否能用书上“孩子滑雪”的例子，说明：什么是online问题？为什么它被认为更困难？

Next Fit Algorithm - NF

- The strategy: Put a new item in the last bin if possible, or use a new bin. Never look back!
- An example: *S*={0.2, 0.5, 0.4, 0.7, 0.1, 0.3, 0.8}



问题9：最多比最优解差多少？

课外作业

- DH: 4.1; 4.2;
- DH: 4.8-4.9; 4.11; 4.12;
- DH: 4.13-14;