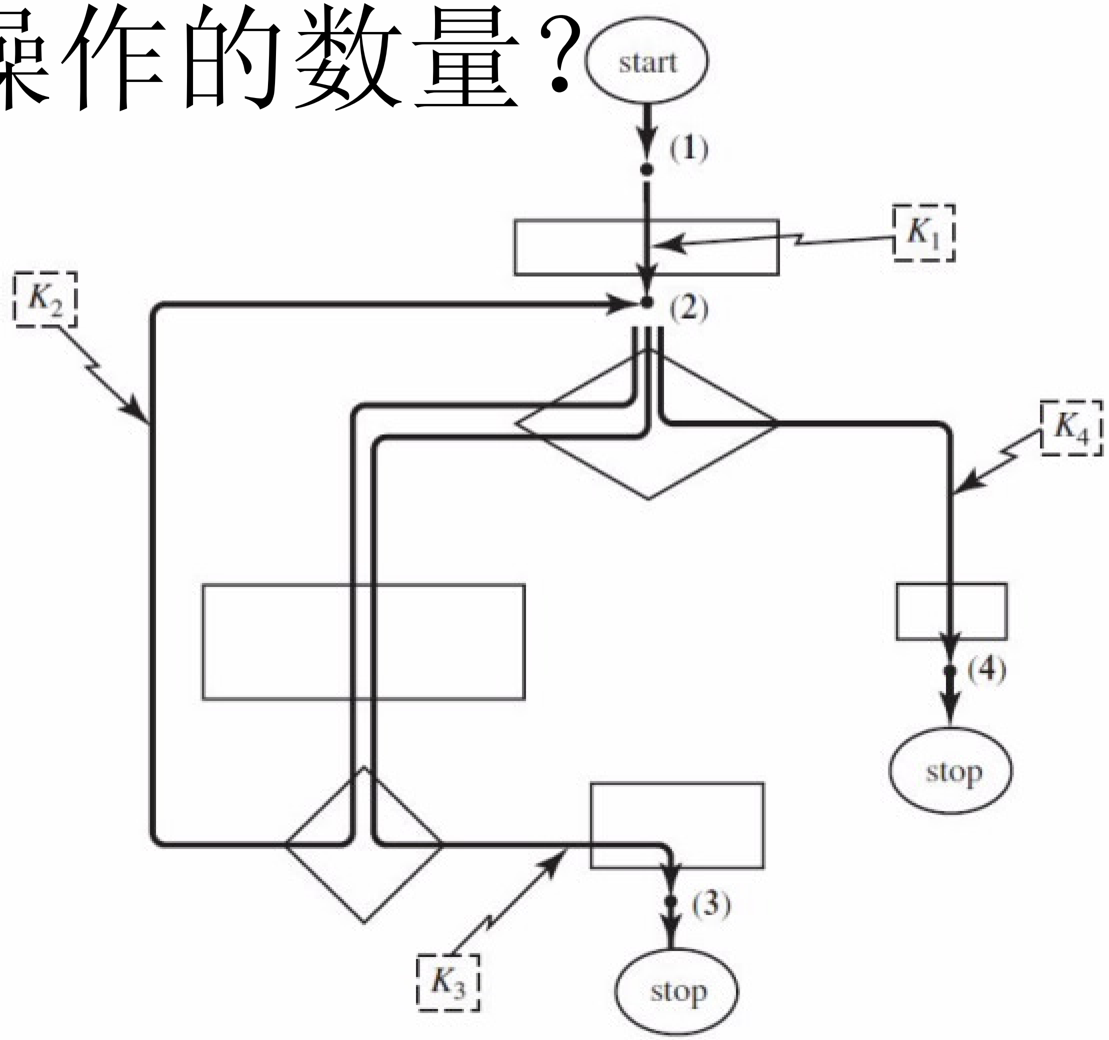
计算机问题求解 – 论题2-04

- 计算机问题和算法

2014年03月11日

问题8：在二分搜索算法的分析中，为什么我们可以只观察“比较”操作的数量？



Worst case

$$\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$$

and

$$\sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$$

(see Appendix A for a review of how to solve these summations), we find that in the worst case, the running time of INSERTION-SORT is

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\ &\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\ &= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right) n \\ &\quad - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

没有必要算这么复杂

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\
&\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\
&= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right)n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right)n \\
&\quad - (c_2 + c_4 + c_5 + c_8)\,.
\end{aligned}
$$

We can express this worst-case running time as $an^2 + bn + c$ for constants a, b, and c that again depend on the statement costs c_i; it is thus a ***quadratic function*** of n.

更进一步：$O(n^2)$

问题9：Big-*O* 的“Robustness”是什么意思？

In other words, as long as the basic set of allowed elementary instructions is agreed on, and as long as any shortcuts taken in high-level descriptions (such as that of Figure 6.1) do not hide unbounded iterations of such instructions, but merely represent finite clusters of them, big-*O* time estimates are *robust*.

问题10：

为什么有时候Big-O可能误导人？

We have known that : $\log n \in o(n^{0.0001})$

$$(\text{since } \lim_{n\to\infty} \frac{\log n}{n^{\varepsilon}} = 0 \ \text{ for any } \varepsilon > 0)$$

However, which is larger : $\log n$ and n^{ε}, if $n = 10^{100}$?

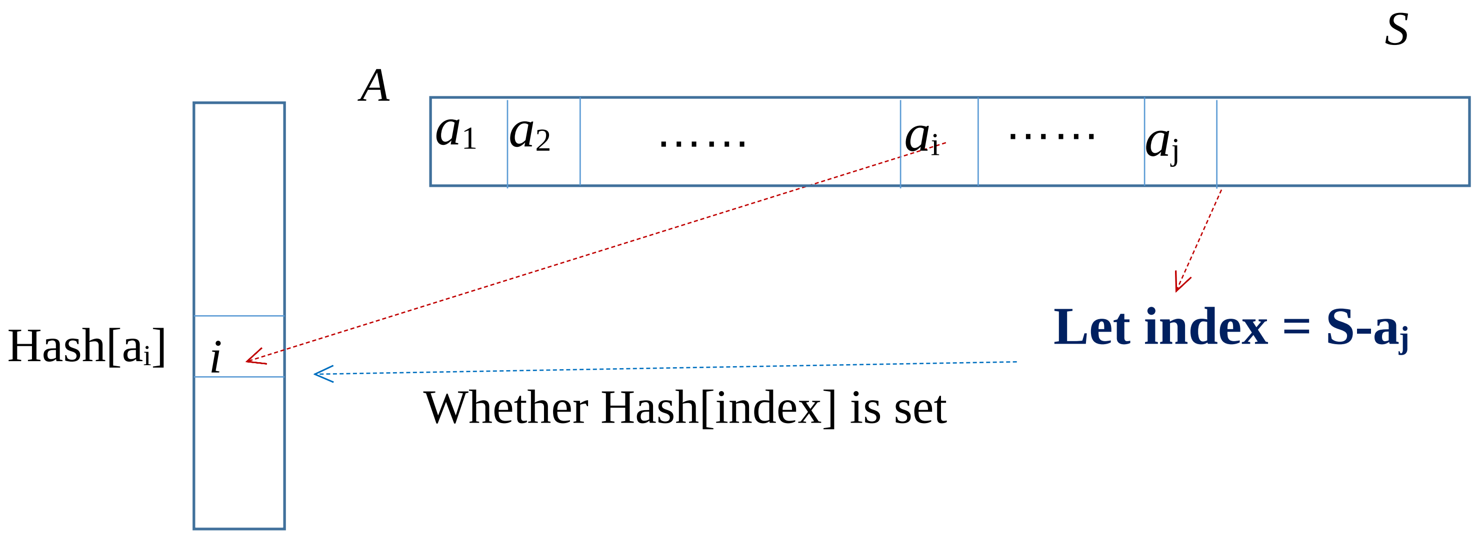
问题10：

从Linear Search到Binary Search, 收益是什么？需要付出什么代价？

考你一下：

Let *A* be an array of integers and *S* a target integer. Design an efficient algorithm for determining if there exist a pair of indices *i*,*j* such that *A*[*i*]+*A*[*j*]=*S*.

如果这里“efficient”是指“线性的”，你的答案满足要求吗？



Sleeping Tigers

(1) find the "lowest" point P_1;

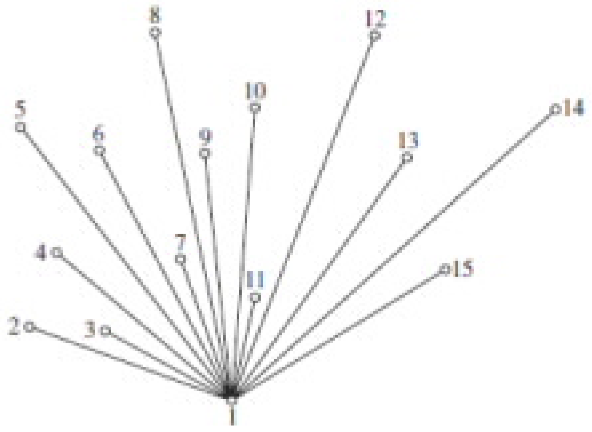
(2) sort the remaining points by the magnitude of the angle they form with the horizontal axis when connected with P_1, and let the resulting list be $P_2, \ldots, P_N$;

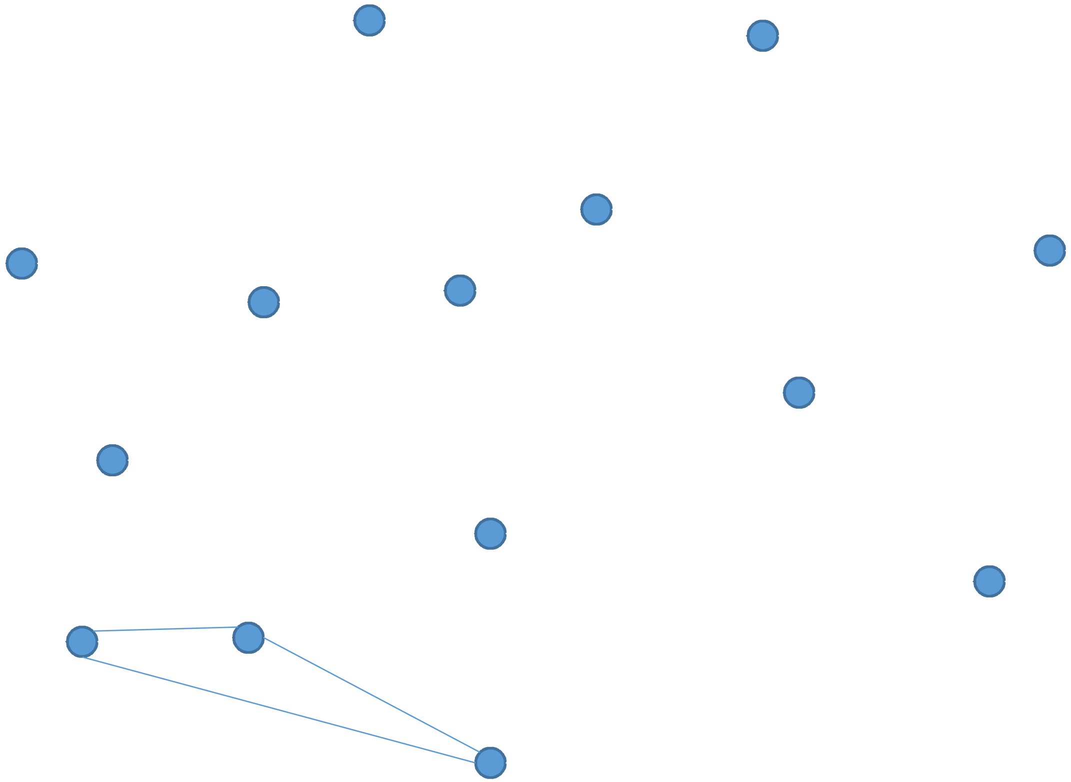
(3) start out with P_1 and P_2 in the current hull;

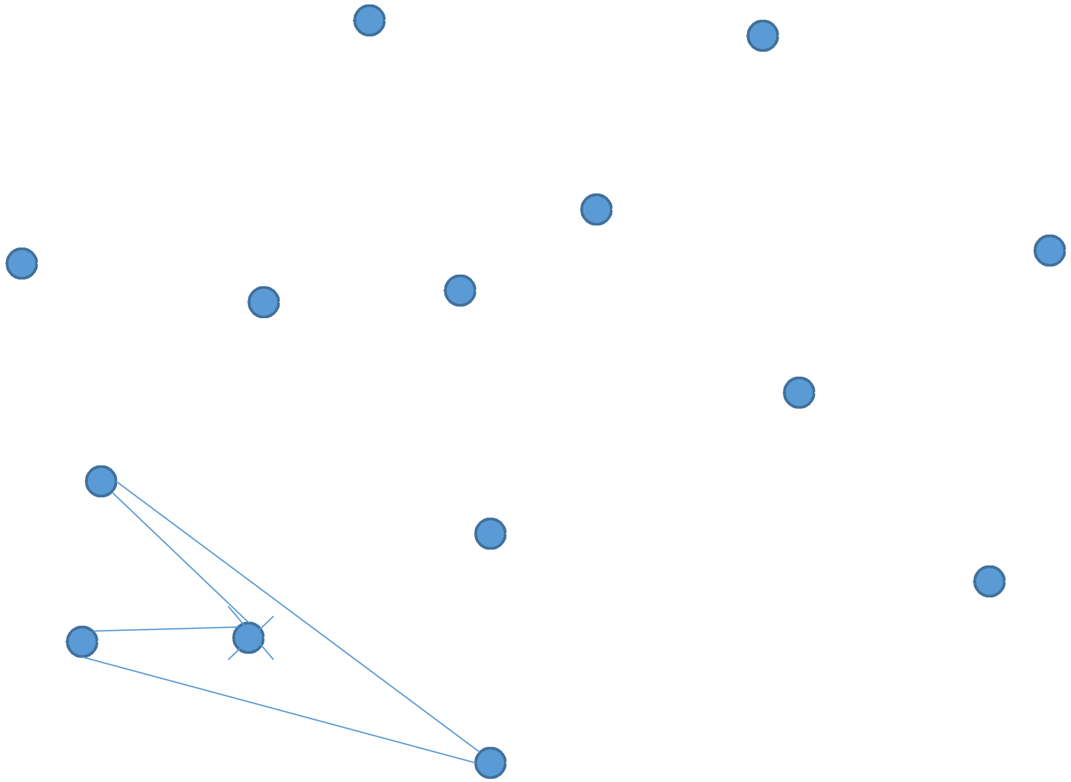
(4) for I from 3 to N do the following:

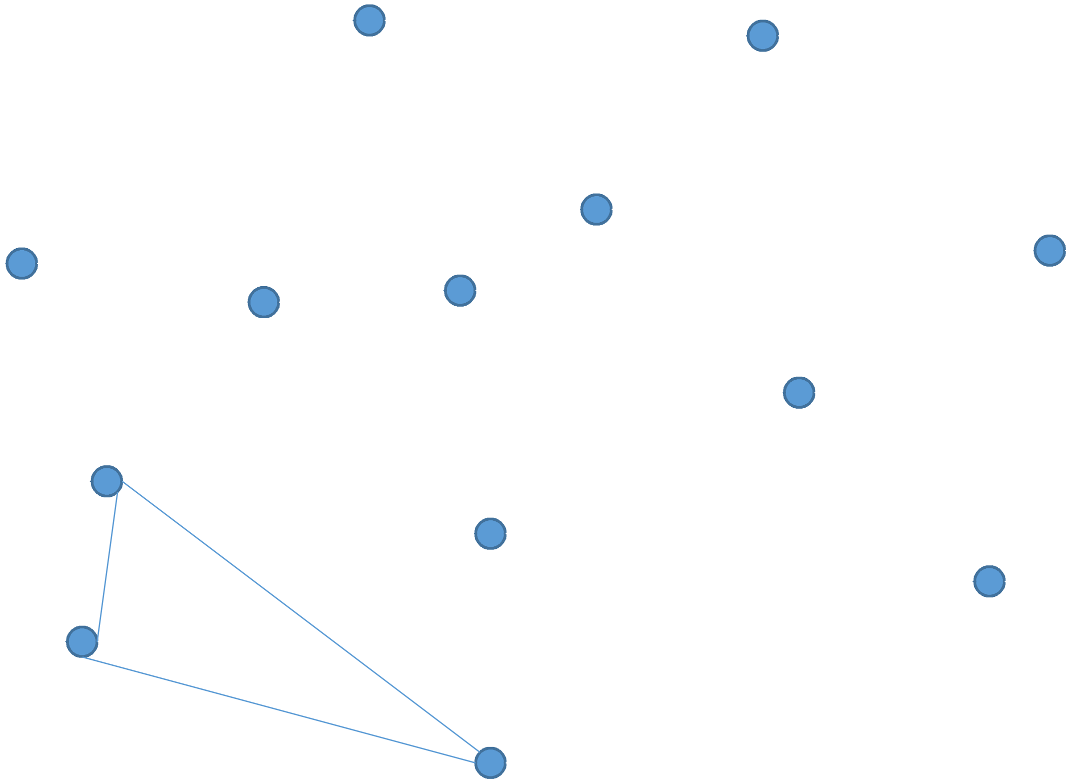
(4.1) add P_I tentatively to the current hull;

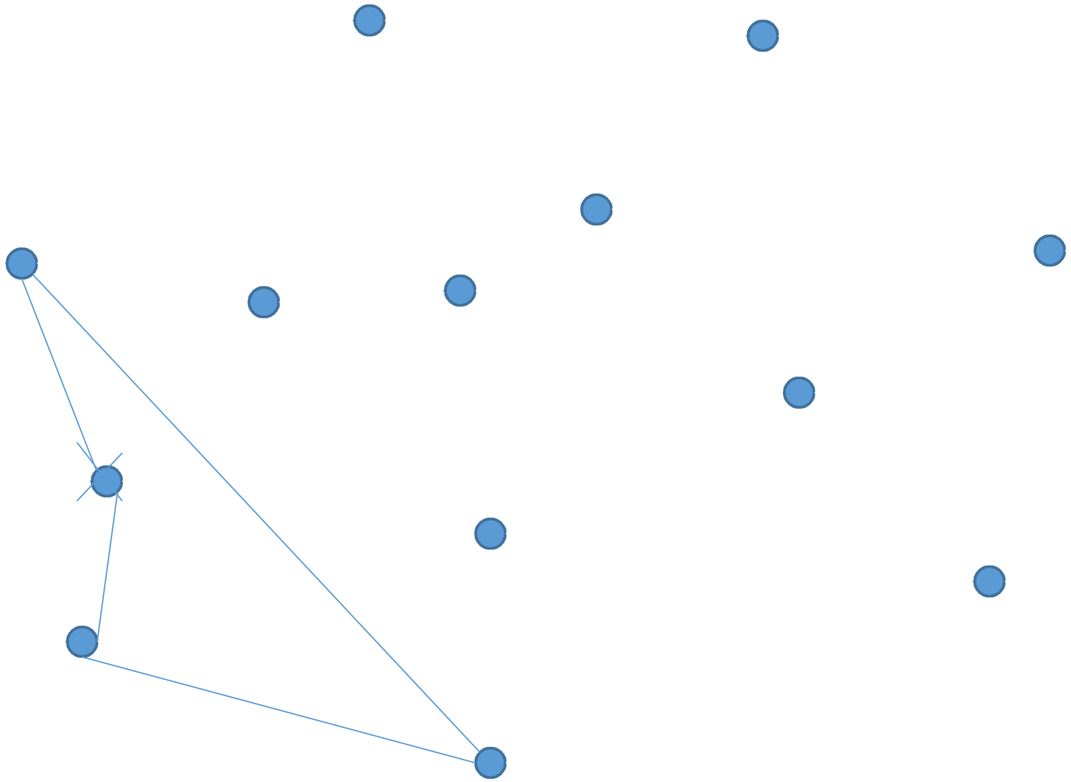
(4.2) work backwards through the current hull, eliminating a point P_J if the two points P_1 and P_I are on different sides of the line between P_{J-1} and P_J, and terminating this backwards scan when a P_J that does not need to be eliminated is encountered.

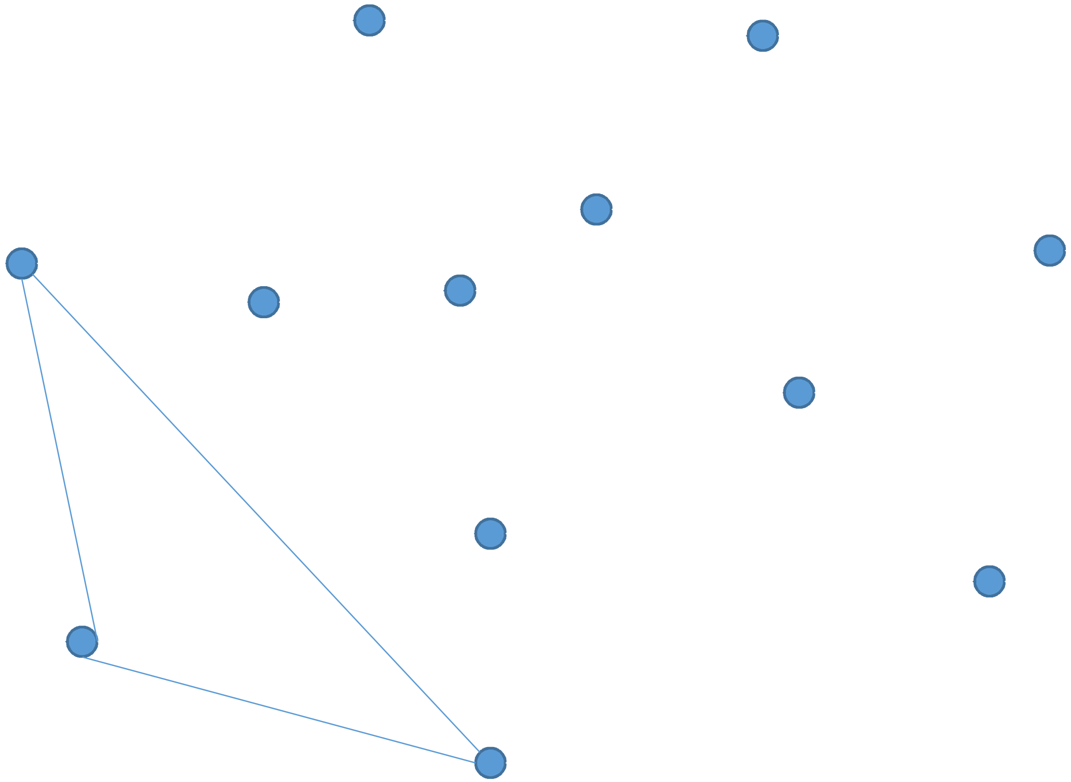


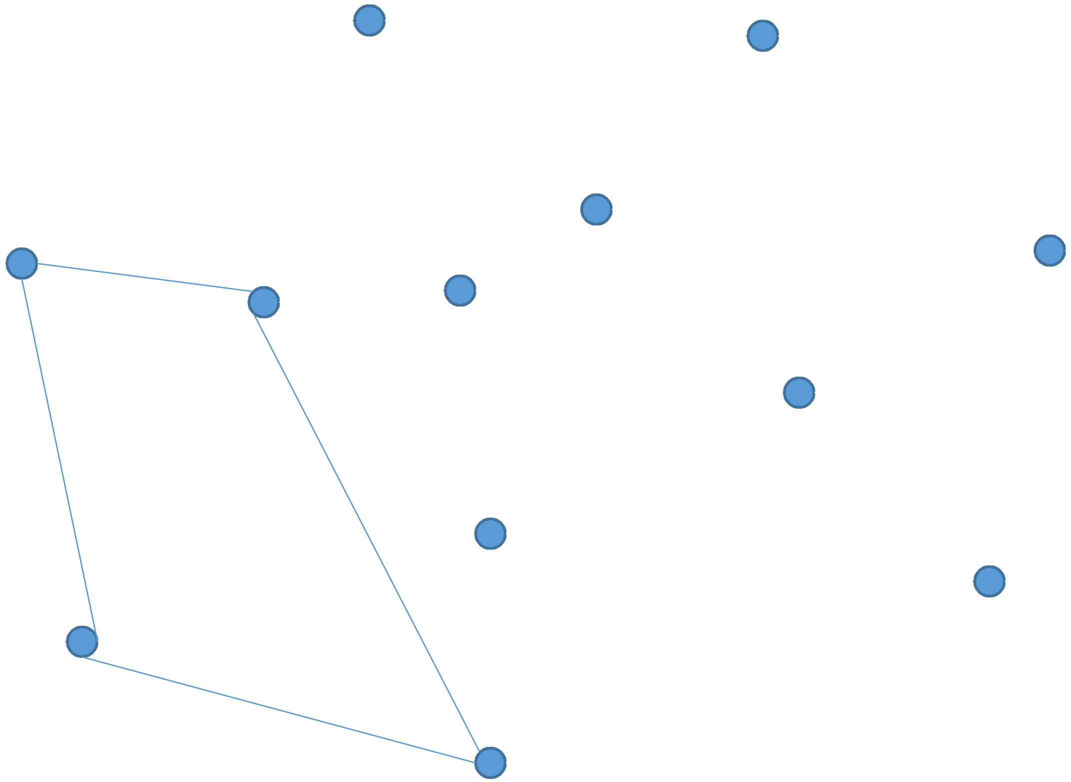


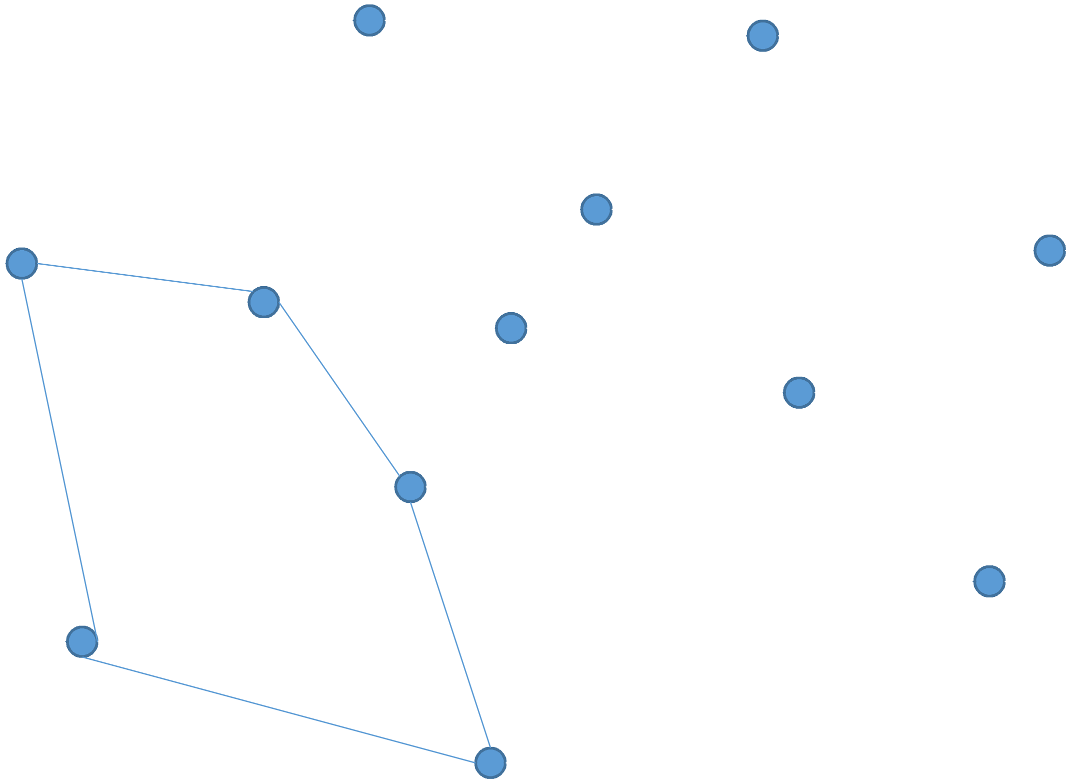


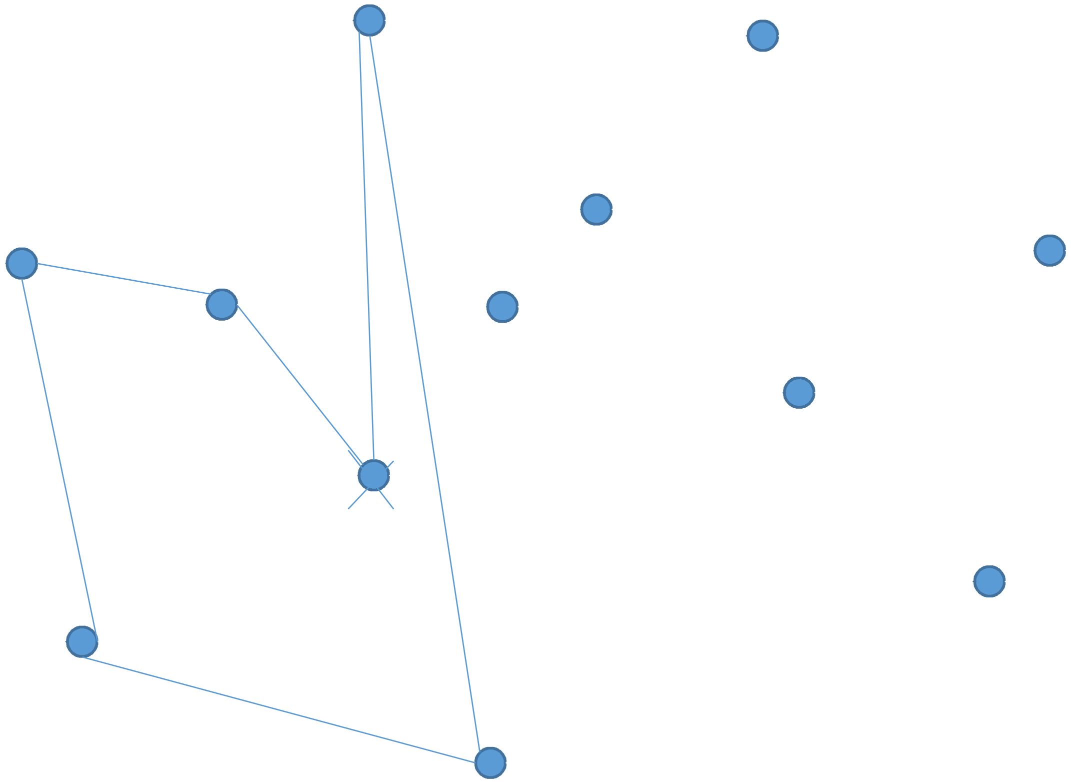


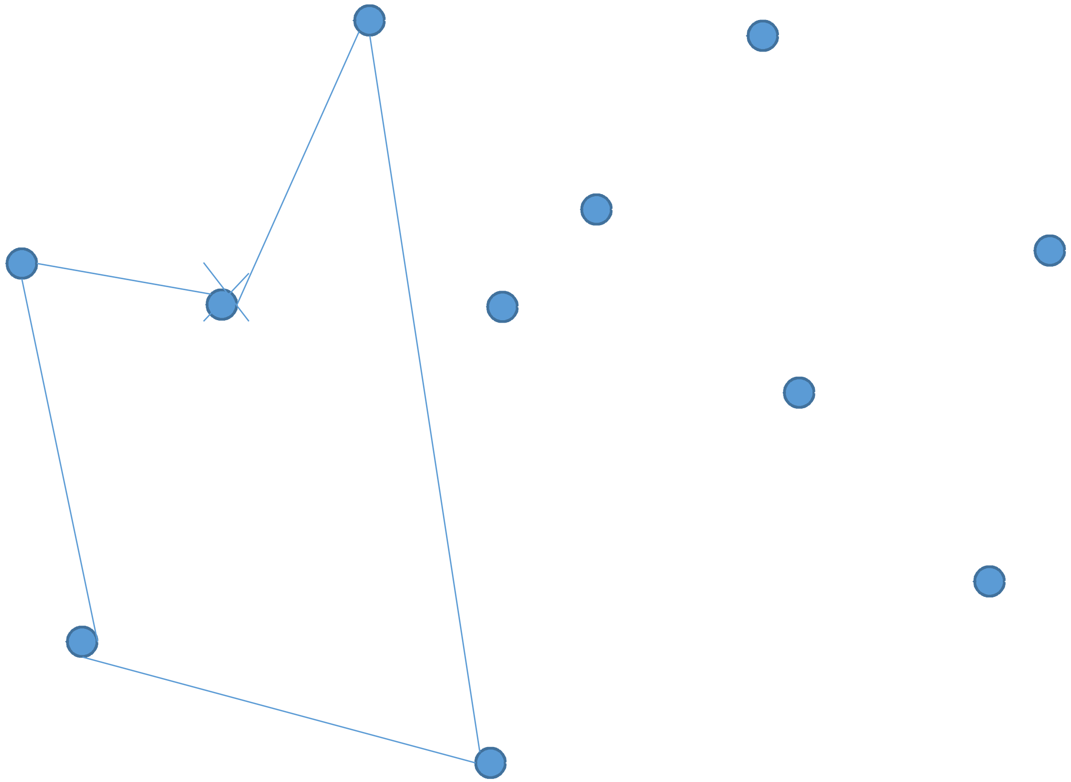












问题11：

其实还有不少非排序算法，其主要代价就是排序

Step (1)	$O(N)$
Step (2)	$O(N \times \log N)$
Step (3)	$O(1)$
Step (4)	$O(N)$
Total	$O(N \times \log N)$

问题1:

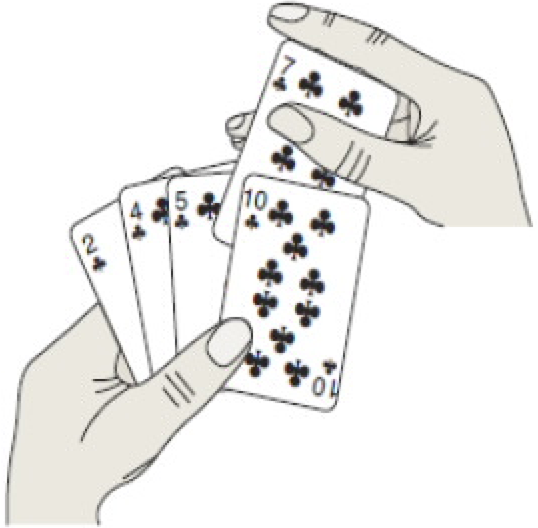
什么是a well-specified computational problem?

算法“实现”输入/输出之间的关系

- 给定一个问题，我们如何讨论一个算法:
  - 基本思路
  - 过程描述
  - 证明其正确
  - 讨论其效率

This is the framework used throughout the courses to think about the design and analysis of algorithms.

基本思路 – 有时非常简单！



问题3：你能否描述一下“插入排序”的基本思路与很多人玩牌时的习惯做法之间的关联？

插入排序过程中的循环不变量

At the start of each iteration of the **for** loop of lines 1–8, the subarray $A[1..j-1]$ consists of the elements originally in $A[1..j-1]$, but in sorted order.

不变是相对于“变”而言的，变的是什么呢？

问题6：

利用loop invariant证明算法正确与用数学归纳法证明数学命题正确有什么异同？

问题9：

你知道为什么插入排序算法的复杂度必然是平方量级的吗？

“逆序”

- 如果输入序列没有重复元素，不妨假设输入就是{1,2,…,n} 的某种排列;
- 所谓“逆序”是指在输入序列中存在一对元素（不一定相邻）： $<x_i, x_j>$ 满足 $x_i>x_j$, but $i<j$；
- 显然，排序的任务就是消除所有的“逆序”。

- 两个相关的问题：
  - 输入中最多可能有多少个“逆序”？
  - 算法中关键运算（比如：比较运算）次数与逆序消除的个数是什么关系?

同样的问题，不同的方法

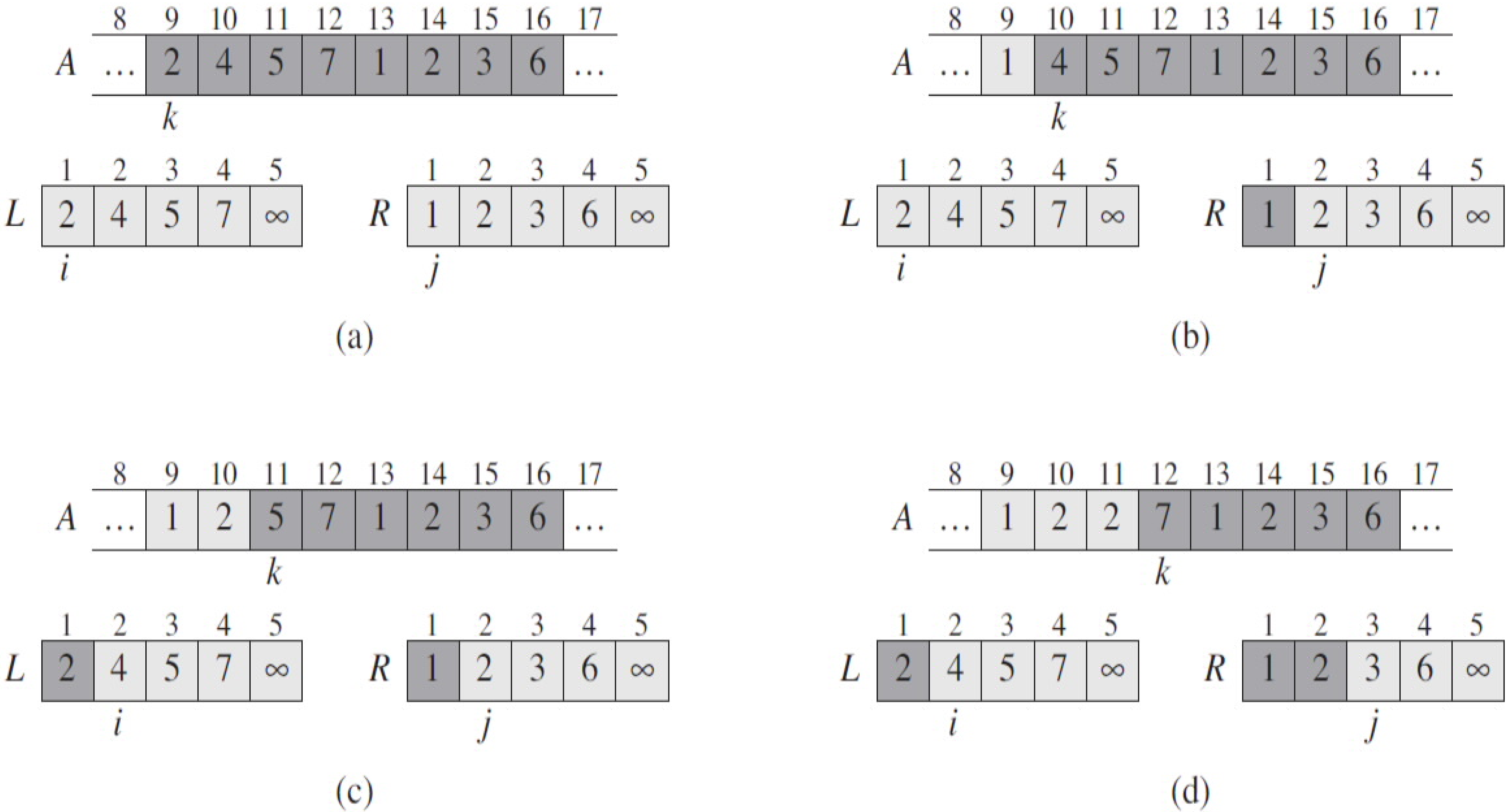
```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

Divide-and-Conquer Recursion

To sort the entire sequence $A = \langle A[1], A[2], \ldots, A[n] \rangle$, we make the initial call MERGE-SORT(A, 1, $A.length$), where once again $A.length = n$.

问题10：

为什么这个算法是正确的？

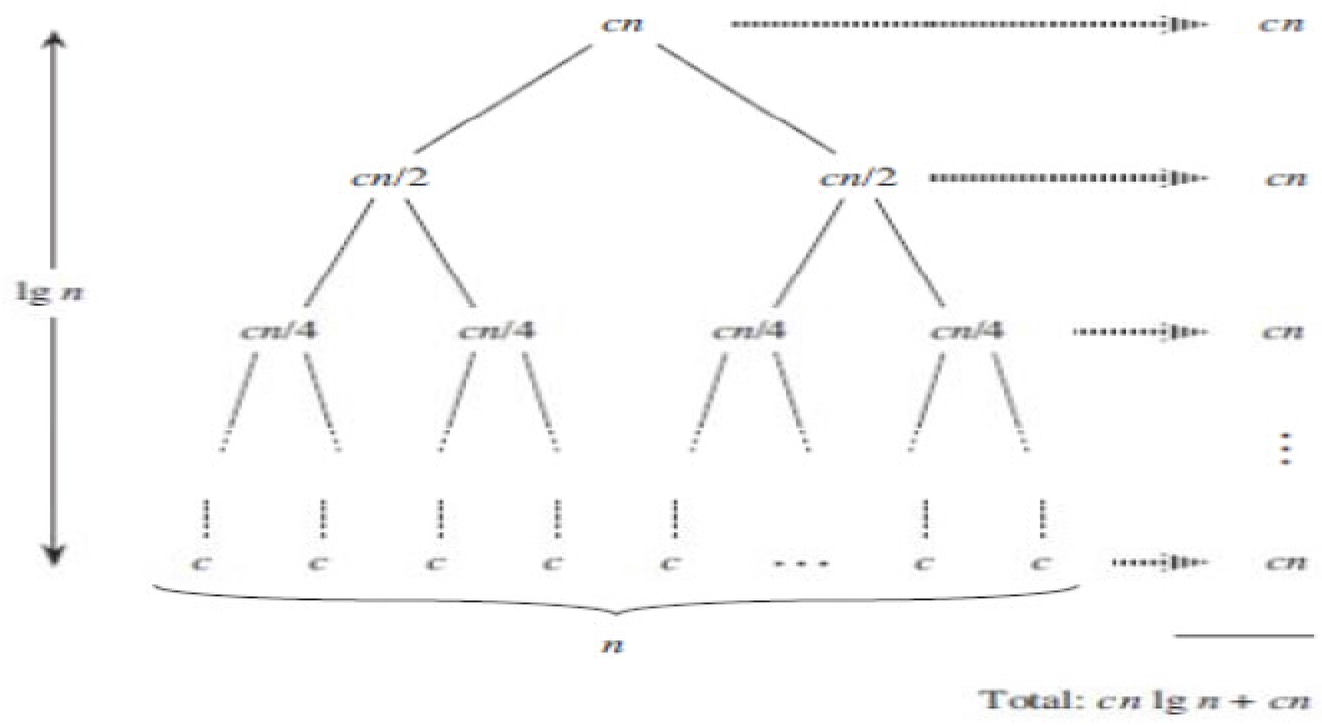


8 9 10 11 12 13 14 15 16 17
A … 2 4 5 7 1 2 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
1 2 3 4 5
R 1 2 3 6 ∞
j
(a)
8 9 10 11 12 13 14 15 16 17
A … 1 4 5 7 1 2 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
1 2 3 4 5
R 1 2 3 6 ∞
j
(b)
8 9 10 11 12 13 14 15 16 17
A … 1 2 5 7 1 2 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
1 2 3 4 5
R 1 2 3 6 ∞
j
(c)
8 9 10 11 12 13 14 15 16 17
A … 1 2 2 7 1 2 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
1 2 3 4 5
R 1 2 3 6 ∞
j
(d)

```
MERGE(A, p, q, r)
1  n1 = q - p + 1
2  n2 = r - q
3  let L[1..n1 + 1] and R[1..n2 + 1] be new arrays
4  for i = 1 to n1
5      L[i] = A[p + i - 1]
6  for j = 1 to n2
7      R[j] = A[q + j]
8  L[n1 + 1] = ∞
9  R[n2 + 1] = ∞
10 i = 1
11 j = 1
12 for k = p to r
13     if L[i] ≤ R[j]
14         A[k] = L[i]
15         i = i + 1
16     else A[k] = R[j]
17         j = j + 1
```

At the start of each iteration of the **for** loop of lines 12–17, the subarray $A[p \,..\, k-1]$ contains the $k - p$ smallest elements of $L[1 \,..\, n_1 + 1]$ and $R[1 \,..\, n_2 + 1]$, in sorted order. Moreover, $L[i]$ and $R[j]$ are the smallest elements of their arrays that have not been copied back into A.

合并排序效率比插入排序高



$O(n\log n)$