

2017

* 符号标注的题目存在争议

选择题

1	2	3	4	5	6	7	8	9	10
C	B	D	A	C	D	C	D	A	C

11	12	13	14	15	16	17	18	19	20
D	B	B	D	B	C	A	D	D	B

21	22	23	24	25	26	27	28	29	30
C	B	D	A	A	A	A	D	B	C

应用题

31

1)

```

1 // 假设每次插入的数都不同
2 void Insert_NewNode(int X) {
3     BinaryNode *tmp = new BinaryNode(X);
4     if (root == NULL) {
5         root = tmp;
6         return;
7     }
8     BinaryNode *p = root;
9     BinaryNode *pre = NULL;
10    while (p != NULL) {
11        pre = p;
12        if (X < p->data) {
13            p->Lsize = p->Lsize + 1;
14            p = p->left;
15        } else {
16            p = p->right;
17        }
18    }
19    if (X < pre->data) {
20        pre->left = tmp;

```

```

21     } else {
22         pre->right = tmp;
23     }
24 }

```

2)

```

1  BinaryNode* Search(int K) {
2      if (root == NULL) return NULL;
3      BinaryNode* p = root;
4      while (true) {
5          if (p->Lsize == K) {
6              break;
7          } else if (p->Lsize > K) {
8              p = p->left;
9          } else {
10             K = K - p->Lsize;
11             p = p->right;
12         }
13     }
14     return p;
15 }

```

3)

平均情况下，搜索树近似于平衡的二叉树，渐进时间复杂度是 $O(\log n)$ ，最坏情况下，搜索树退化成线性链表，渐进时间复杂度是 $O(n)$ 。

32

1)

通常选择比较操作作为代表操作，因为排序本质是消除逆序，消除逆序的前提是元素比较。

2)

采用相邻元素比较的排序算法，一次只能消除一个逆序对，而快排和归并算法能够利用之前的比较信息，一次消除多个逆序对，所以有更高的性能。

3)

采用分治算法。将待求序列分成 p_1 、 p_2 两段（ p_1 在前， p_2 在后），假设 p_1 、 p_2 已经排好序并计算出内部的逆序对数（否则在 p_1 、 p_2 上重复算法，直到只有单个元素），先比较两者的第一个元素 a 和 b 。

1. 如果 a 比 b 小，则将 a 复制到临时数组的首位， p_1 下一次比较元素推进一位
2. 否则 b 比 a 小，将 b 复制到临时数组的首位， p_2 下一次比较元素推进一位，并且总逆序数（ p_1 、 p_2 的逆序之和）增加 p_1 中比 b 大的元素个数（即从 a 到 p_1 结尾）
3. 临时数组的待放位置也推进一位。

p_1 、 p_2 有一方没有可比较元素后，将另一方剩下的元素直接复制到临时数组的剩下位置。

算法的递推公式是 $T(n) = 2T(n/2) + c * n$ ，由 master 定理可知算法的复杂度是 $O(n \log n)$ 。

33

1)

用图上的节点表示蓄水池，节点之间的有向边表示单向通道。供水问题转化为，给定有向图，寻找能到达其余所有点的节点。

2)

```
1 struct Edge {
2     int dest;
3     Edge * next;
4 };
5
6 struct graph {
7     vector<Edge*> first; // first out edge
8     vector<string> vName; // vertex name
9 } G;
```

3)

```
1 vector<bool> visited(n);
2 void dfs(int id) {
3     visited[id] = true;
4     for (auto e : first) {
5         if (!visited[e->dest]) {
6             dfs(e->dest);
7         }
8     }
9 }
10 bool isAccessable(int start) {
11     dfs(start);
12     for (auto flag : visited) {
13         if (!flag) return false;
14     } // if there exists node unvisited after dfs, then start node
15         // can not access all nodes;
16     return true;
17 }
```

4)

先利用 tarjon 算法计算出图中的连通片，收缩连通片后的图是无环的，若存在孤立连通片，如果多于一个则原问题无解，如果只有一个孤立点则原图是强连通的，所有点都满足要求；在收缩图上利用拓扑排序可以计算出所有没有边到达的点（由于无环，这样的点必然存在），若这样的点超过一个则原问题无解，否则所求的连通片内的点都恰好能满足要求。

tarjon 算法可以一遍得到所有的连通片，复杂度是 $O(n + m)$ ，求拓扑序时，是在更小的图上遍历的，复杂度是 $O(n' + m')$ ，所以总的时间复杂度是 $O(n + m)$ 。

34

思路：不公平的读写问题

```
1 semaphore mutex = 1;
2 semaphore wnd = 1;
3 int cnt = 0;
4
5 vip_custom() {
6     进入大厅
7     P(mutex);
8     cnt++;
9     if (cnt == 1)
10         P(wnd);
11     V(mutex);
12     得到服务
```

```

13     P(mutex);
14     cnt--;
15     if (cnt == 0)
16         V(wnd);
17     V(mutex);
18     离开
19 }
20
21 simple_custom() {
22     进入大厅
23     P(wnd);
24     得到服务
25     V(wnd);
26     离开
27 }

```

35

略

36

1)

目录文件最多占用 $8 \times 1\text{KB} = 8\text{KB}$ 的物理空间（每个直接索引指向一页）

每个目录项占 32B，故最多登记 $8\text{KB}/32\text{B} = 256$ 个目录项

即每个目录下包含的文件和子目录数的卢纶最大值是256

2)

每一物理页可以登记 $1\text{KB}/4\text{B} = 256$ 个索引项，故文件理论最大尺寸为

$(8 + 256 + 256 * 256) * 1\text{KB} = 65800\text{KB}$

*3)

n级索引项指向由n-1级索引项组成的集合（本答案约定叫n级索引页或块），直接索引是0级，就是物理块的地址。

除了直接和一级索引所包含的物理空间，还需要 $266 - 8 - 256 = 2\text{KB}$ 的物理空间

所以还需要用到二级索引页的一条一级所应项所指索引页中的2个条目（直接索引）。

索引块所占空间为 $32\text{B} + (256 + 256 + 256) * 4\text{B} = 3104\text{B}$ （10个索引项包含在文件目录项中，有32B，对于二级索引块，虽然只使用了一部分条目，但是系统分配时是直接分配一个物理块的）

所以共占磁盘空间 $266\text{KB} + 3104\text{B} = 269.03125\text{KB}$

37

1)

栈从高地址想低地址增长

2)

funct 函数入口处

EBP = 0xbc000030

ESP = 0xbc000020

3)

第5、6两行指令用于将参数 `&y` 传入栈帧中，传入的参数内容是 `0xbc000018`，存放参数的地址是 `0xbbffffffc`

4)

将返回地址压栈，更新 EIP 到 `scanf` 代码的首地址。

5)

返回值存放在寄存器 `%eax`

38

略

39

略（见自定向下相关章节习题，原题）
