一、选择题（7*5=35）

1、优先队列分支限界法解旅行售货员问题时，活节点表的组织形式是_____。 （B）

A、最大堆　　B、最小堆　　C、栈　　D、队列

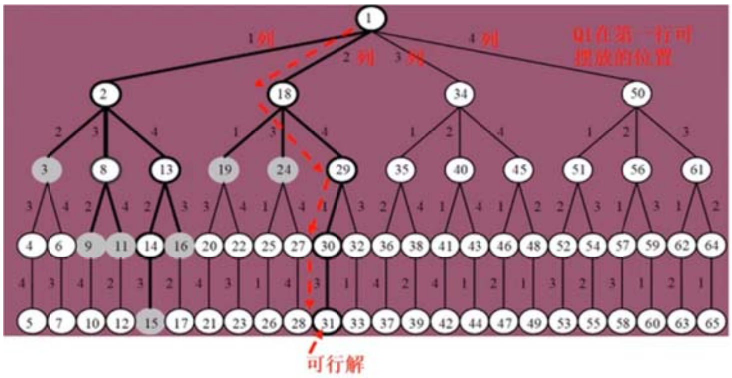
解析：TSP 问题求解最小值，因此为最小堆

2、一个 6 皇后问题，其解空间的深度为_______，其中第一层为根节点。 （D）

A、8　　B、5　　C、6　　D、7

解析：如四皇后问题的解空间树如下所示：

“四皇后”问题的解空
间树是一个完全4叉树的
简化，如下图所示。



皇后 N=4，第一层为根节点，因此解空间的深度为 N+1

若第 0 层为根节点，则深度为 N

详见网页：

https://blog.csdn.net/weixin_40113704/article/details/89509128

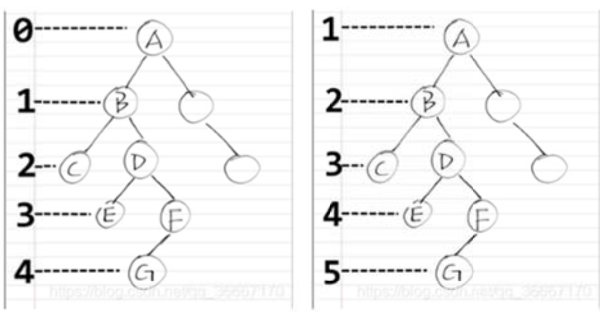


图	左	右
层数	从第0层开始	从第1层开始
最大层数	4	5
深度	4	5
高度（高度=深度）	4	5
高度（数层数）	5	5

3、回溯法求解旅行售货员问题是的解空间树是_______。（B）

A、子集树　　　　　　　　　　B、排列树

C、深度优先生成树　　　　　　D、广度优先生产树

4、$2^n \times 2^n$的棋盘覆盖问题的时间复杂性是（）。（D）

A. n^2　　B. n^4　　C. 2^n　　D. 4^n

解析：见 PPT：

棋盘覆盖算法的复杂性

- 设f(k)是棋盘覆盖算法覆盖$2^k \times 2^k$的棋盘所需要的时间，则f(k)满足如下递归方程：

$$f(k) = \begin{cases} O(1) & k = 0 \\ 4f(k-1) + O(1) & k > 0 \end{cases}$$

可知$f(k) = O(4^k)$。

Cha.3 递归与分治 P85 每次都是把一个棋盘分成 4 份，然后调整为 4 个同样的问题之后再计算。

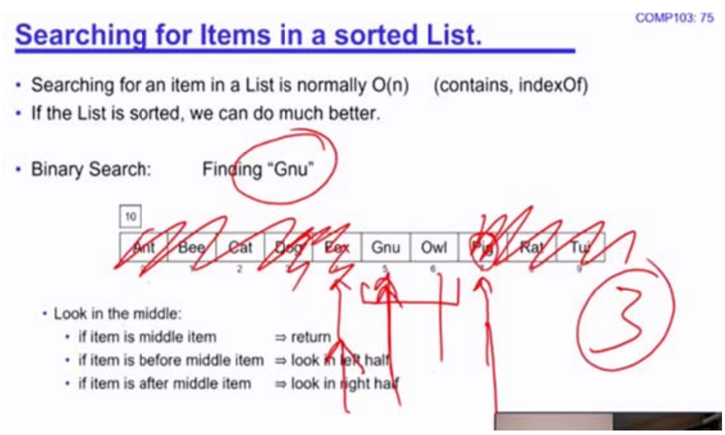
求解方法：先写出递归方程，再把四个选项代入，验证即可求得为 D：　$4^k=4*4^{k-1}$

若是 n×n 的棋盘，则 $f(n)=O(n^2)$

5、二分搜索算法是利用（ ）实现的算法。（A）

A. 分治法　B. 动态规划　C. 贪心算法　D. 回溯法

解析：



Binary Search (recursive)

Sorted List

$[low, high)$

```
public int indexOf(String value, List<String> data){
    return indexOf(value, data, 0, data.size());
}

public int indexOf(String value, List<String> data, int low, int high){
    // value in [low .. high)  (if present)
    if (low >= high){ return -1; }                                  //  value not present
    int mid  =  (low + high) / 2;
    int comp =  value.compareTo(data.get(mid));
    if (comp == 0)       { return mid;  }                           // item is present
    else if (comp < 0)   { return indexOf(value, data, low, mid); } // item in [low .. mid)
    else                 { return indexOf(value, data, mid+1, high);} // item in [mid+1 .. high)
}
```


Binary Search (recursive)

$n \quad n/2 - n/4 - n/8 - n/16$

Cost:

- each recursive call cuts the range in half.
- number of recursive calls = number of times can cut n items in half = $\log_2(n)$
- cost of each line (except recursive calls) = O(1)
- Total cost = O(log(n))

```
public int indexOf(String value, List<String> data, int low, int high){
    // value in [low .. high)  (if present)
    if (low >= high){ return -1; }                                  //  value not present
    int mid  =  (low + high) / 2;
    int comp =  value.compareTo(data.get(mid));
    if (comp == 0)       { return mid;  }                           // item is present
    else if (comp < 0)   { return indexOf(value, data, low, mid); } // item in [low .. mid)
    else                 { return indexOf(value, data, mid+1, high);} // item in [mid+1 .. high)
}
```


Binary Search (recursive)

1,000,000 20

Cost:

- each recursive call cuts the range in half.
- number of recursive calls = number of times can cut n items in half = $\log_2(n)$
- cost of each line (except recursive calls) = O(1)
- Total cost = O(log(n))

$\log_2(1000) \approx 10$

$\log_2(n) \times O(1)$

$O(\lg n)$

```
public int indexOf(String value, List<String> data, int low, int high){
    // value in [low .. high)  (if present)
    if (low >= high){ return -1; }                                  //  value not present
    int mid  =  (low + high) / 2;
    int comp =  value.compareTo(data.get(mid));
    if (comp == 0)       { return mid;  }                           // item is present
    else if (comp < 0)   { return indexOf(value, data, low, mid); } // item in [low .. mid)
    else                 { return indexOf(value, data, mid+1, high);} // item in [mid+1 .. high)
}
```

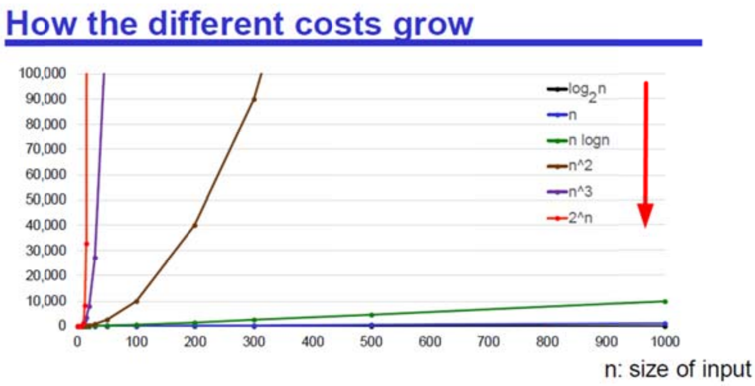
6、在下列算法中通常以自底向上的方式求解最优解的是（ ）。（B）

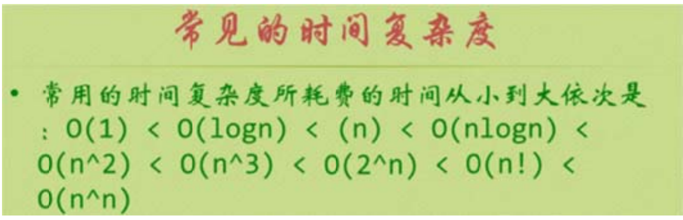
A. 备忘录法　B. 动态规划　C. 贪心算法　D. 回溯法

7、下列时间复杂性中，最大的是（ ）。　（C）

A. n　B. $\log^n$　C. n^2　D. $n\log^n$

解析：

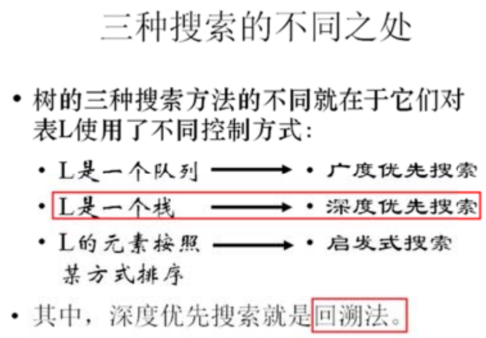




二、填空题（5*5=25）

1、以深度优先方式系统搜索问题解的算法称为____（1）回溯___，它使用的数据结构一般是____（2）栈___。

解析：见 PPT：

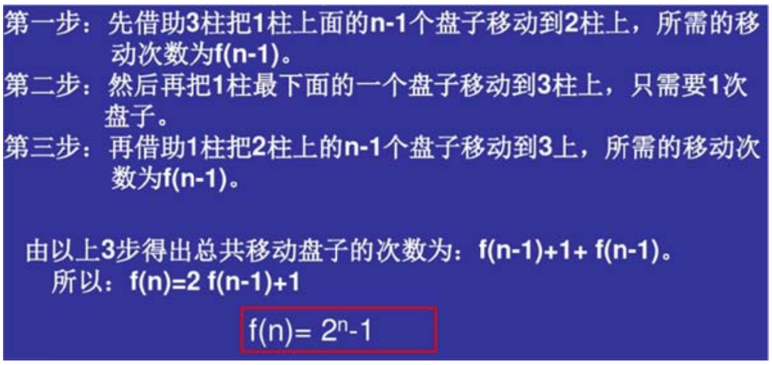


2、已知某算法的时间复杂度估计如下：

$$f(n)=\begin{cases}1 & n=1\\ 2f(n-1)+1 & n>1\end{cases}$$

则，该算法的时间复杂度为__(3)__ _$o(2^n)$_。

解析：考查的是汉诺塔问题的时间复杂度：



https://wenku.baidu.com/view/e4c160f882d049649b6648d7c1c708a1294a0a1c.html

$2^n-1=2*(2^{n-1}-1)+1=2^n-2+1=2^n-1$

可参见课本 P47 “代入法求递推式”

3、一般采用自底向上的方式求解最优解的是（4 动态规划）。

4、甲在纸上写下了一个 100 以内的正整数让乙猜，乙每猜一次数，甲都给出一个提示“太大”或“太小”，这样，乙在不考虑运气的前提下至少要猜多少次可确定此数。（5）7 次

解析：$\log_2 100$ 向上取整=7

三、算法题（20+15=35）

1、0-1 背包问题：给定 n 种物品和一背包。物品 i 的重量是 w_i>0,其价值为 v_i >0,背包容量为 C >0, 1≤i≤n。要求找一 n 元向量$(x_1,x_2,\ldots,x_n)$, $x_i\in\{0,1\}$, $\sum w_ix_i\le C$,且$\sum v_ix_i$达最大。问应如何选择装入背包中的物品，使得装入背包中物品的总价值最大？

设 0-1 背包问题的最优值为 *m(i, j)*，即 *m(i, j)*是背包容量为 j，可选择物品为 1，2，…, i，时 0-1 背包问题的最优值。请完成以下题目。

（1）证明最优子结构性质。（5 分）

（2）写出 *m(i, j)*的递归关系和边界式子（状态转移方程）。（5 分）

（3）有 5 个物品，其重量分别是{2, 1, 5, 4, 3}，价值分别为{9, 5, 7,3, 5}，背包的容量为 10。以下是 *m(i, j)*表，请完成(a)~(e)填空，并给出最终选择的物品。（10 分）

	0	1	2	3	4	5	6	7	8	9	10
0											
1											
2									(d)		
3									(c)		

4									(b)		
5									(a)		(e)

解析：

（1）证明最优子结构性质：

证明：假设(x1,x2,……,xn)是容量为 c 的背包的一组最优解，其中 xi 的取值为 0 或 1，表示是否放入背包中。则必有（x2,x3,……,xn）为如下子问题的一组最优解：

sum{xi*wi} $_{(2<=i<=n)}$<=c-x1*w1

利用反证法证明，假设（y1,y2,……,yn）是该子问题的一组最优解而（x2,x3,……,xn）不是。则

sum{yi*vi} > sum{xi*vi}　(2<=i<=n)

那么就可得到 ：

x1*v1+ sum{yi*vi} > x1*v1+ sum{xi*vi}　(2<=i<=n)

则(x1,y2,……,yn)是原问题的最优解，而(x1,x2,……,xn)不是，与假设“(x1,x2,……,xn)是容量为 c 的背包的一组最优解”矛盾。因此 0-1 背包具有最优子结构性质。

（2）**算法思想：**令背包的载重量范围为 $0 \sim m$ 。

m[i][j]表示前 i 个物体中，能够装入载重量为 j 的背包中的物体的最大价值，$j = 1, 2, ..., m$ 。

可以得到下面的动态规划函数：

$$optp_i(0) = optp_0(j) = 0 \qquad (2.1)$$

$$optp_i(j) = \begin{cases} optp_{i-1}(j) & j < w_i \\ \max\{optp_{i-1}(j), optp_{i-1}(j - w_i) + p_i\} & j \geq w_i \end{cases} \qquad (2.2)$$

状态转移方程：

$$m[i][j] = \begin{cases} 0 & i=0 \text{ or } j=0 \\ m[i-1][j] & j<wi \\ \max\{m[i-1][j-w[i]]+v[i], m[i-1][j]\} & j>=wi \end{cases}$$

在 装第 i 个物品 与 不装第 i 个物品 中选择大者

证明：

m[i][j]表示在物品数为 i，背包容量为 j 的情况下所得到的最大价值总和。当物品数为 0 或背包容量为 0 的时候，最大价值自然为 0；当物品数量增加到第 i 个的时候，若背包容量 j 比 wi 小，则无法装入该物品，因此物品 i 并未起到作用，相当于没有物品 i，则 m[i][j]=m[i-1][j];若背包容量 j 比 wi 大，则要比较加入物品 i 和不加入物品 i 这两种情况下哪种方案的价值总和最大，即

m[i][j]= max{m[i-1][j-w[i]]+v[i],m[i-1][j]}.

装物品 i 的价值为何是 m[i-1][j-w[i]]　+　v[i]？　因为已经装了 i 之后重 j，因此 j-w[i]为对应的前 i-1 个物品的重量，所以 m[i-1][j-w[i]]对应装前 i-1 个物品的最大价值，加上第 i 个物品的价值，即为装物品 i 后的最大价值。

（3）有 5 个物品，其重量分别是{2, 1, 5, 4, 3}，价值分别为{9, 5, 7,3, 5}，背包的容量为 10。以下是 *m(i, j)*表，请完成(a)~(e)填空，并给出最终选择的物品。（10 分）

解析：即

物品编号	1	2	3	4	5
物品重量	2	1	5	4	3
物品价值	9	5	7	3	5

背包容量

物品编号

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	9	9	9	9	9	9	9	9	9
2	0	5	9	14	**14**	14	**14**	14	14	14	14
3	0	5	9	14	**14**	14	14	**16**	21	21	21
4	0	5	9	14	14	14	14	17	21	21	21
5	0	**5**	9	14	14	14	19	19	21	21	22

以蓝色那行，即物品编号 1 对应的那一行为例：可供选择的物品只有第 1 个
以此类推：物品标号 0、1、2、3、4、5 对应的各行可供选择的物品分别是
无物品、前一个物品、前两个物品、前三个物品、前四个物品、前五个物品，即

0	无物品（循环初始）
1	前一个物品（编号 1 物品）
2	前两个物品（编号 1 和 2 物品）
3	前三个物品（编号 1、2 和 3 物品）
4	前四个物品（编号 1、2、3 和 4 物品）
5	前五个物品（编号 1、2、3、4 和 5 物品）

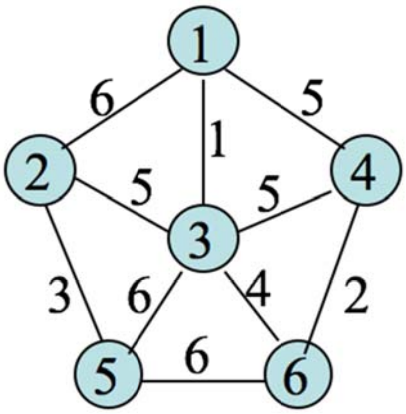
以 17 为例 说明更新规律：填这个格子时 是和它正上方的 16 比，这个格子可以选择装物品编号 4，因此只需要比较装了 4 和不装 4（对应 16）哪个大即可，装了 4，则还有重量 7-（物品 4 的重量）4=3 来装前面的物品，通过重量 3 对应的价值格子 14 找到 3 重量最大的价值为 14，因此装了 4 后，对应的价值即为 14+（物品 4 的价值）3=17，比没装 4 的价值 16 大，因此当候选可装的物品为 1、2、3、4 时，最大价值为 17.

再说明 22 装的是哪几个物品？倒推：和上一行的 21 不一样，因此装了物品编号 5，还剩

10-3=7 个重量，重量 7 对应的列中，不算物品 5 最大价值 17 和上一行的 16 价值不同，因此装了物品编号 4，还剩 7-4=3 个重量，重量 3 对应的列中，不算物品 4、5，最大价值 14，和 9 不一样，因此装了物品 2，还剩 3-1=2 个重量，重量 2 对应列中，不算物品 2 到 5，最大价值 9 和上一行的 0 不同，因此装了物品编号 1，

因此装了物品编号 1 2 4 5 最大价值为 22.

2、利用 kruskal 算法计算如下图的最小生成树。（15 分）



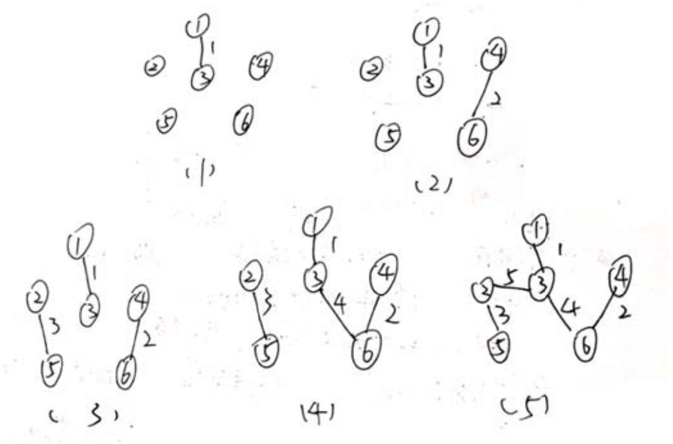
要求：

（1）、画出联通分支的变化状态。

（2）、写出边的选择情况。

（3）、证明该算法具有贪心选择性质。

解析：（1）



（2）kruskal 贪心策略：

基本思想：在保证无回路的前提下依次选出权重较小的 n－1 条边（n 为结点数）。

具体做法：若(i, j)是 E 中尚未被选的边中权重最小的，且(i, j)不会与已经选择的边构成回路，于是就选择 (i, j)。

i	j	w
1	3	1
4	6	2
2	5	3
3	6	4
2	3	5

（3）证明 kruskal 算法的正确性：

贪心选择性质

- 若某问题的整体最优解可通过一系列局部最优解的选择，即贪心选择，来达到，则称此问题具有贪心选择性质。
- 贪心选择每次选取当前最优解，它依赖于以往的选择，即要与以往选择相容，而与以后的选择无关。
- 对于满足贪心选择性质的问题，采用贪心算法是能够获得最优解的。

如何确定贪心选择性质

- 确定问题具有贪心选择性质的方法：
- (1)证明第一个贪心选择是正确的，即它必定包含在某个整体最优解之中。
- (2)证明在做了第一个贪心选择后，原问题便简化为规模较小的相同子问题，即可继续使用贪心选择。
- 证明了以上两点后，由数学归纳法可知该问题具有贪心选择性质。

因此，按照上述步骤，得出证明过程为：

（1）证明第一个选择是对的：按照 Kruskal 算法，第一个选择就是权值最小的边，记为 e。不妨设 e 连接顶点 i 和 j。假设存在一个最小生成树 H，且 H 不包含 e，即顶点 i 和 j 是联通的，但不直接相连。那么我们用 e 去替换顶点 i 和 j 这个联通分支中的某条边 e1，显然，w(e)<=w(e1)。则新生成的生成树 H’ 会比 H 的权值更小，因为 w(H’)=w(H)-w(e1)+w(e)<= w(H)。与 H 是最小生成树矛盾，所以 e 一定在最优生成树中。

（2）把 G 中的顶点 i 和 j 看作一个独立的联通分支，去掉权值最小的边 e，得到 G’。

显然把 G’ 的最优生成树中加入和边 e 便得到 G 的最优生成树。这样原问题便转化为求 G’ 的最优生成树的问题。

（3）由数学归纳法可知该问题具有贪心选择性质。

因此采用该贪心算法可以获得最优解，即原命题得证。

3、给定任务集及截止时间和误时惩罚如下：（5 分）

i	1	2	3	4	5	6	7
d[i]	4	2	4	3	1	4	6
w[i]	70	60	50	40	30	20	10

请给出最后的任务时间安排表，和最小的误时惩罚（使用贪心算法）。

解析：

任务时间表：

4	2	3	1	5	7	6

5、6 未安排，因此最小误时惩罚为 30+20=50

证明贪心算法的正确性即为证明问题具有贪心选择性质：

任务时间表的贪心选择性质

- 我们先来看第一个选择是否是正确的。
- 设任务i的误时惩罚wi是最大的。若任务i不在任何一个最优时间表内，则任取一个最优时间表P，其中任务i未能按时完成。
- 修改P为P’：将任务i取代某个任务j，使任务i能按时完成，而任务j未能按时完成。
- ∵ $wi \geq wj$ ∴ $W(P') = W(P) - wi + wj \leq W(P)$。
- 矛盾。故第一个选择必在最优时间表内。

因为任务已经按照误时惩罚由大到小排序

因此第一个选择即为任务 i

假设 P 已经是最优安排 W（P）最小，但又找到另一个安排 P’的误时惩罚 W（P’）比 W（P）更小，所以与假设矛盾！

- 已知首选正确。再来看首选之后，原问题是否转化为规模较小的相同问题。
- 首选后，对剩余n – 1个任务依然要最优安排，才能与首选构成n个任务的最优安排。
- 首选后，此问题便转化为对剩余n – 1个任务的最优时间表问题。
- 所以，对此贪心选择策略，任务时间表问题具有贪心选择性质。