**算法设计与分析 实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 高嘉 | | 院系 | 软件学院 | | 学号 | 2021211948 |
| 任课教师 | | 王金宝 | | | 指导教师 | 王金宝 | |
| 实验地点 | | 研究院 | | | 实验时间 | 2023年 4月 27 日 | |
| 实验名称 | | 树搜索算法设计与实现 | | | | | |
| 同 组 人 | | 无 | | | | | |
| 实验内容 | | | | | | | |
| **1、问题**   * **活动选择问题**   1. 输入：*S={1, 2, …, n}*, *F*={ [*si，fi*] }，*n≥i≥1*   输出：*S*中的最大相容活动集合  要求:  （1）使用动态规划技术求解活动选择问题. 请写出你使用的递归方程，说明其中各项的含义; 请说明优化子结构以及问题是否具有子问题重叠性; 实现你设计的算法.）  （2）使用贪心算法求解活动选择问题；（实现课件上的算法即可）  （3）使用树搜索算法求解活动选择问题；（说明如何将问题转化为树搜索问题；说明数中每个节点的含义；说明你使用的分支界限搜索算法）  （随机生成若干活动，并使用它测试动态规划算法、贪心算法和树搜索算法的运行时间和活动数量）  **2、基于动态规划技术的算法**  **递归方程：**  d[f[i]]=max(d[f[i]],d[s[i]-1]+1)  **优化子结构：**  按结束时间从小到大排序，若选择了第一个任务，那S1={2,3,4，…,n}的最优解也是S最优解去掉第一个任务的最优解。设从时间s[2]到f[n]最优有k个，那如果f[1]<s[2]，S的最优解就应有k+1个，若f[1]>=s[2]，那么S最优解就有k个。  **算法代码：**  sort(f+1,f+1+n);  for(int i=1;i<=n;i++)  {  d[f[i]]=max(d[f[i]],d[s[i]-1]+1);  for(int j=f[i];j<=f[n];j++)  if(d[j]<d[f[i]]) d[j]=d[f[i]];  }  printf("%d",d[f[n]]);  **3、贪心算法**  算法代码：  sort(f+1,f+1+n);  int ans=1;  for(int i=2;i<=n;i++)  if(f[i-1]<s[i]) ans++;  printf("%d",ans);    **4、树搜索算法**  **节点含义：**  先按f[i]从小到大排序，再把第一个点作为根节点，根节点的任务数为1，向右为选该任务，向左为不选该任务，每个节点记录此时选取的任务数。  **分支界限：**  因为只能从小到大按顺序选择，所以若此时选择了k，那该情况下最大的方案数是(n-k)+num，若此时最大方案数也小于ans，可知该分支不能取得最好方案，所以直接返回即可。  **算法代码：**  主函数：  sort(f+1,f+1+n);  dfs(1,1,f[1]);  printf("%d",ans);  dfs函数：  void dfs(int x,int num,int final)  {  if(x==n) {ans=max(ans,num);return;}  if(num+(n-x)<ans) return;//就算后面全选也不可能是最大值  if(s[x+1]>final) dfs(x+1,num+1,f[x+1]);//选  dfs(x+1,num,final); //不选  } | | | | | | | |
| 实验结论（结果分析、遇到的困难和解决方法等） | | | | | | 备注 |  |
| 结果分析：  该算法中，贪心算法效率最高，动态规划次之，树搜索最慢。 | | | | | | | |