**北京邮电大学软件学院**

**2016－2017学年第一学期实验报告**

**课程名称： 算法设计与分析**

**项目名称： 分治法与贪心法**

**项目完成人：**

**姓名：\_\_肖逸敏\_\_\_\_\_\_学号：\_\_2014211990\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**指导教师：\_\_李朝晖\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**日 期： 2016 年 12 月 1 日**

1. **实验目的**

1、 深刻理解并掌握分治法、贪心法的设计思想；

2、 提高应用分治法、贪心法的技能

1. **实验内容**

基本题 1：元素选择问题 给定序列中 n 个元素和一个整数 k，1≤k≤n，请用分治法输出这 n 个元 素中第 k 小元素的值及其位置。编写并调试程序。

基本题 2：0—1 背包问题 给定 n 种物品和一背包。物品 i 的重量是 wi，其价值为 vi，背包的容量 为 C。请用贪心法（按单价最高的贪心策略）问应如何选择装入背包的物品， 使得装入背包中物品的总价值最大?

提高题 1：最大子段和问题 给定由 n 个整数组成的序列(a1, a2, …, an)，请用分治法求解最大子段和 问题，要求该序列求解和为最大的连续子段。

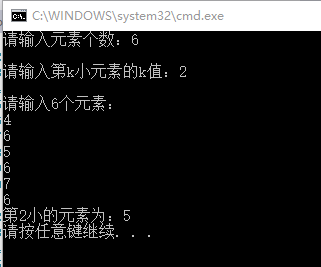
提高题 2：区间调度问题（见课件，用贪心法）

1. **实验环境**

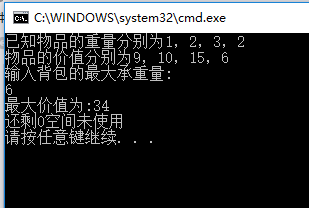
vs2013，C++

1. **实验结果**

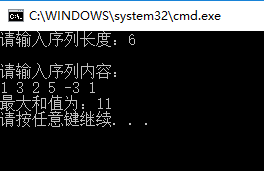
基础题1：



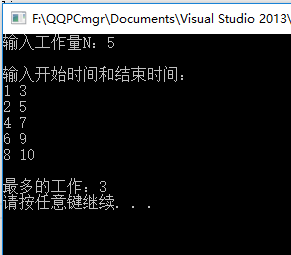
基础题2：



提高题1：



提高题2:



1. **附录**

基础题1：

分析：给出一组数据，返回第i小的元素，用分治法解决，用到之前快排里的一个函数partition，为了把数组分成两部分，A[p,q-1]的元素都小于主元q， A[q+1,r]的元素都大于q， 所以只要比较i这个位置要落在哪一边就好了

#include<iostream>

#include<cstdio>

#include<algorithm>

using namespace std;

int Partition(int \*A, int p, int r)

{

int x = A[r];

int i = p - 1;

for (int j = p; j <= r - 1; j++)

{

if (A[j] <= x)

{

i++;

swap(A[i], A[j]);

}

}

swap(A[i + 1], A[r]);

return i + 1;

}

int Select(int \*A, int p, int r, int i)

{

if (p == r)

return A[p];

int q = Partition(A, p, r);

int k = q - p + 1;

if (i == k)

return A[q];

else if (i<k)

return Select(A, p, q - 1, i);

else

return Select(A, q + 1, r, i - k);

}

int A[30000];

int main()

{

int n, ii;

cout << "请输入元素个数：";

cin >> n;

cout<<endl;

cout << "请输入第k小元素的k值：";

cin >> ii;

cout<< endl;

cout << "请输入"<<n<<"个元素："<<endl;

for (int i = 1; i <= n; i++)

cin >> A[i];

int ans = Select(A, 1, n, ii);

cout <<"第"<<ii<<"小的元素为："<< ans << endl;

}

基础题2：

分析：贪心原则为单位价值最大且重量最小，不超过背包最大承重量为约束条件。也就是说，存在单位重量价值相等的两个包，则选取重量较小的那个背包。

具体实现过程是：首先可以设置一个备份pvu类型的数组，在不破环原数据的情况下，对此备份数组按单位重量价值从大到小的排序。依次设立两个指针i，j(其中i表示当前应该参与最佳pv值的元素指针，j表示符合约束条件的指针（单位重量价值PV最大，重量最小，不超过最大承重量约束)

#include <iostream>

using namespace std;

typedef struct

{

int v;

int w;

float pv;

}pvu;

void sortByPv(pvu[], int);

int zeroneBags(pvu[], int, int, int \*);

void print(pvu a[], int n)

{

for (int i = 0; i<n; i++)

{

cout << a[i].w << " " << a[i].v << " " << a[i].pv << endl;

}

cout << endl;

}

int main() {

int remained=0;

int i, maxw;

int w[] = { 1, 2, 3, 2 };

int v[] = { 9, 10, 15, 6 };

int n = sizeof(w) / sizeof(int);

int N = n;

pvu arr[30];

for (i = 0; i<n; i++)

{

arr[i].v = v[i];

arr[i].w = w[i];

arr[i].pv = v[i] \* 1.0 / w[i];

}

cout << "已知物品的重量分别为1，2，3，2" << endl;

cout << "物品的价值分别为9，10，15，6" << endl;

cout << "输入背包的最大承重量:\n";

cin >> maxw;

cout << "最大价值为:" << zeroneBags(arr, n, maxw, &remained) << "\n还剩" << remained << "空间未使用" << endl;

return 0;

}

void sortByPv(pvu arr[], int n)

{

pvu t;

int i, j;

for (i = 0; i<n - 1; i++)

for (j = 0; j<n - 1 - i; j++)

if (arr[j].pv<arr[j + 1].pv)

{

t = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = t;

}

}

int zeroneBags(pvu arr[], int n, int maxw, int \*e)

{

int i = 0, j, minw, totalv = 0;

int avail = maxw;

sortByPv(arr, n); //按最大单位重量价值PV从大到小的排序

while (avail&&i<n)

{

minw=i;

for (j=0;j<n;j++)

if (arr[i].pv==arr[j].pv)

{

if (arr[i].w>arr[j].w&&j>i)

{

minw=j;

}

}

if (arr[minw].w<=avail)

{

avail-=arr[minw].w;

totalv+=arr[minw].v;

i++;

}

else

i++;

}

\*e = avail;

return totalv;

}

提高题1：

分析：将该序列平分为两段（A1..An/2;An/2….An），最大字段段和有三种情况，1：为左边的最大子段和leftsum，2：为右边的最大字段和rightsum，3：由左边靠近中点的最大连续子段+右边靠近中点的最大连续子段midsum。所以最终结果为三者中的最大值（MAX（leftsum,rightsum,midsum））。

#include <iostream>

using namespace std;

int MaxSum(int a[], int left, int right);

int main()

{

int n, a[100];

cout << "请输入序列长度：";

cin >> n;

cout << endl;

cout << "请输入序列内容：" << endl;

for (int i = 0; i<n; i++)

cin >> a[i];

int sum = MaxSum(a, 0, n - 1);

cout << "最大和值为：";

cout << sum << endl;

return 0;

}

int MaxSum(int a[], int left, int right)

{

int sum = 0, midsum = 0, leftsum = 0, rightsum = 0;

int center, s1, s2, lefts, rights;

if (left == right)

sum = a[left];

else

{

center = (left + right) / 2;

leftsum = MaxSum(a, left, center);

rightsum = MaxSum(a, center + 1, right);

s1 = -9999, lefts = 0;

for (int j = center; j >= left; j--)

{

lefts += a[j];

if (lefts>s1)

s1 = lefts;

}

s2 = -9999, rights = 0;

for (int i = center + 1; i <= right; i++)

{

rights += a[i];

if (rights>s2)

s2 = rights;

}

midsum = s1 + s2;

if (midsum<leftsum)

sum = leftsum;

else

sum = midsum;

if (sum<rightsum)

sum = rightsum;

}

return sum;

}

提高题2：

分析：问题描述：有N项工作，每项工作有开始时间si和结束时间ti，让你选择最多的工作，工作之间不存在时间交叉。

按结束时间最早排序才是正确的。

1.因为结束时间最早，所以取相同数量的工作肯定它最早结束或至少不晚。

2.如果后面还有工作，那么加入到结束最早算法的空间更大。

因此就不会存在比它更多工作的可能。

#include<iostream>

#include<algorithm>

using namespace std;

const int MAX = 100000;

int N, s[MAX], t[MAX];//输入，s开始时间，t结束时间

pair<int, int> itv[MAX];//用于对工作排序的pair数组

int main()

{

cout << "输入工作量N：";

cin >> N;

cout << endl;

cout << "输入开始时间和结束时间：" << endl;

int i;

for (i = 0; i<N; i++)

cin >> itv[i].second >> itv[i].first;//便于以结束时间排序，所以改存顺序

cout << endl;

sort(itv, itv + N);

int ans = 0, t = 0;

for (i = 0; i < N; i++){

if (t < itv[i].second){

ans++;

t = itv[i].first;

}

}

cout << "最多的工作：";

cout << ans << endl;

system("pause");

return 0;

}