САА - Упражнение 4 (Лаб\_4\_САА)

VIII.Алчни (Greedy) алгоритми

Задача 32

Програма за плащане с най-малък брой монети чрез алчен алгоритъм (итеративен вариант).

#include "stdafx.h"

#include <iostream>

using namespace std;

const int br = 6;

int moneti[br] = {50,20,10,5,2,1};

void stotinki(int suma)

{

int i, j;

j = suma;

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

{

cout << "Broi moneti po " << moneti[i] << " st. = " << j/moneti[i]<<endl;

j = j % moneti[i];

}

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int sum;

cout << "Enter the number: ";

cin >> sum;

stotinki(sum);

return 0;

}

Задача 33\*

Съставете алгоритъм и напишете програма за плащане с най-малък брой монети чрез алчен алгоритъм (рекурсивен вариант).

IX.Разделяй и Владей

Задача 34

Програма за изчисляване на най-голям общ делител на две естествени числа чрез рекурсия.

#include "stdafx.h"

#include <complex>

#include <iostream>

using namespace std;

int min(int p, int q)

{

if (p > q)

return q;

else

return p;

}

int nod(int c, int d)

{

if( c == d)

return c;

else

return nod(abs(c-d), min(c,d));

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int a, b;

cout << "a = ";

cin >> a;

cout << "b = ";

cin >> b;

cout << "NOD = " << nod(a,b) << "\n";

return 0;

}

Задача 35

Задача за Ханойските кули.

#include "stdafx.h"

#include <iostream>

using namespace std;

const unsigned N = 3;

void diskMove(unsigned N, char a, char b)

{

cout << N << a << b << endl;

}

void hanoy(char a, char c, char b, unsigned numb)

{

if (1 == numb)

diskMove(1, a, c);

else

{

hanoy(a, b, c, numb-1);

diskMove(numb, a, c);

hanoy(b, c, a, numb-1);

}

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

cout << N << endl;

hanoy('A', 'C', 'B', N);

return 0;

}

X.Динамично програмиране

Задача 36

Програма за изчисляване на числото на Фибоначи по въведен номер в редицата на Фибоначи чрез рекурсия.

#include "stdafx.h"

#include <iostream>

using namespace std;

int fib(int n)

{

if (n == 0)

return 0;

else if (n == 1)

return 1;

else

return fib(n-1) + fib(n-2);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int n;

cout << "Fn: ";

cin >> n;

cout << "nF: " << fib(n) << "\n";

return 0;

}

Задача 37

Програма за изчисляване на числото на Фибоначи по въведен номер в редицата на Фибоначи чрез рекурсия и динамично програмиране.

#include "stdafx.h"

#include <iostream>

using namespace std;

#define N 100

unsigned long mas[N];

unsigned long fib(int n)

{

if(n<2)

mas[n] = n;

else if(0 == mas[n])

mas[n] = fib(n-1) + fib(n-2);

return mas[n];

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int n;

cout << "Fn: ";

cin >> n;

cout << "nF: " << fib(n) << "\n";

return 0;

}

Задача 38

Задача за раницата.

#include "stdafx.h"

#include <iostream>

#include <fstream>

#include <iomanip>

using namespace std;

#define MAXN 30 /\* max items\*/

#define MAXM 1000 /\* max capacity \*/

char set[MAXM][MAXN];

/\* set[i][j]==1 means that for capacity i an optimal solution contains item j \*/

unsigned int Fn[MAXM]; /\* objective function \*/

/\* items \*/

/\* No. 1 2 3 4 5 6 7 8 \*/

unsigned int m[MAXN] = {0, 30, 15, 50, 10, 20, 40, 5, 65}; /\* volume \*/

unsigned int c[MAXN] = {0, 5, 3, 9, 1, 2, 7, 1, 12}; /\* price \*/

unsigned int M = 70; /\* actual capacity \*/

unsigned int N = 8; /\* number of items \*/

void calculate()

{ unsigned int maxValue, maxIndex, i, j;

memset(set, 0, sizeof(set)); /\* set[i][j] = 0 \*/

for (i=1; i<=M; i++)

{ maxValue = maxIndex = 0;

for (j=1; j<=N; j++) /\* try the item j \*/

if (m[j]<=i && !set[i-m[j]][j])

if (c[j] + Fn[i-m[j]] > maxValue)

{ maxValue = c[j] + Fn[i-m[j]];

maxIndex = j;

}

if (maxIndex > 0) /\* succesful ! \*/

{ Fn[i] = maxValue;

memcpy(set[i], set[i-m[maxIndex]], N);

set[i][maxIndex] = 1;

}

if (Fn[i] < Fn[i-1])

{ Fn[i] = Fn[i-1];

memcpy(set[i], set[i-1], N);

}

}

}

void write()

{ for (int i=1; i<=N; i++)

if (set[M][i]) cout << setw(4) << i << " ";

cout << endl;

cout << "Max value: " << Fn[M] << "\n";

}

void read()

{

cin >> M >> N;

for (int i=1; i<=N; i++) cin >> m[i];

for (int i=1; i<=N; i++) cin >> c[i];

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

// read();

calculate();

write();

return 0;

}

XI.Сортиране

Задача 39

Метод на мехурчето (Sort\_Bubble).

#include "stdafx.h"

#include <iostream>

#define N 5

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

int i, j, temp, a[N];

cout << "Enter the elements of one-dimensional array\n";

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

{

cout << "a[" << i << "] = ";

cin >> a[i];

}

for(i=N-1; i>=0; i--)

{

for(j=1; j<=i; j++)

{

if(a[j-1] > a[j])

{

temp = a[j-1];

a[j-1] = a[j];

a[j] = temp;

}

}

}

cout << "\nThe sorted elements of one-dimensional array\n";

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

cout << "a[" << i << "] = " << a[i] << "\n";

return 0;

}

Задача 40

Сортиране чрез вмъкване (Sort\_Insertion).

#include "stdafx.h"

#include <iostream>

#define N 5

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

int a[N], i, j, index;

cout << "Enter the elements of one-dimensional array\n";

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

{

cout << "a[" << i << "] = ";

cin >> a[i];

}

for(i=1; i<N; i++)

{

index = a[i];

j = i;

while((j>0) && (a[j-1]>index))

{

a[j] = a[j-1];

j = j - 1;

}

a[j] = index;

}

cout << "\nThe sorted elements of one-dimensional array\n";

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

cout << "a[" << i << "] = " << a[i] << "\n";

return 0;

}

Задача 41

Селективна сортировка (Sort\_Selection).

#include "stdafx.h"

#include <iostream>

#define N 5

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

int i, j, a[N], min, imin;

cout << "Enter the elements of one-dimensional array\n";

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

{

cout << "a[" << i << "] = ";

cin >> a[i];

}

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

{

min = a[i];

imin = i;

for(j=i+1; j<N; j++)

{

if(min>a[j])

{

min = a[j];

imin = j;

}

}

a[imin] = a[i];

a[i] = min;

}

cout << "\nThe sorted elements of one-dimensional array\n";

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

cout << "a[" << i << "] = " << a[i] << "\n";

return 0;

}

Задача 42

Бърза сортировка (Sort\_Quick).

#include "stdafx.h"

#include <iostream>

#define N 5

using namespace std;

int a[N];

void quicksort (int lo, int hi)

{

int i = lo, j = hi, h;

int x = a[(lo+hi)/2];

do

{

while (a[i]<x) i++;

while (a[j]>x) j--;

if (i<=j)

{

h = a[i];

a[i] = a[j];

a[j] = h;

i++; j--;

}

}while (i<=j);

if (lo<j)

quicksort(lo, j);

if (i<hi)

quicksort(i, hi);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int i;

cout << "Enter the elements of one-dimensional array\n";

for(i=0; i<5; i++)

{

cout << "a[" << i << "] = ";

cin >> a[i];

}

quicksort(0, 4);

cout << "\nThe sorted elements of one-dimensional array\n";

for(i=0; i<5; i++)

cout << "a[" << i << "] = " << a[i] << "\n";

return 0;

}

Задача 43

Сортировка на Шел (Sort\_Shell).

#include "stdafx.h"

#include <iostream>

using namespace std;

#define N 5

int \_tmain(int argc, \_TCHAR\* argv[])

{

int i, j, increment, temp, a[N];

cout << "Enter the elements of one-dimensional array\n";

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

{

cout << "a[" << i << "] = ";

cin >> a[i];

}

increment = 4;

while (increment > 0)

{

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

{

j = i;

temp = a[i];

while ((j >= increment) && (a[j-increment] > temp))

{

a[j] = a[j - increment];

j = j - increment;

}

a[j] = temp;

}

if (increment/2 != 0)

increment = increment/2;

else if (increment == 1)

increment = 0;

else

increment = 1;

}

cout << "\nThe sorted elements of one-dimensional array\n";

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

cout << "a[" << i << "] = " << a[i] << "\n";

return 0;

}

Задача 44

Пирамидална сортировка (Sort\_Heap).

#include "stdafx.h"

#include <iostream>

using namespace std;

#define N 5

int a[N];

void siftDown(int root, int bottom)

{

int done, maxChild, temp;

done = 0;

while ((root\*2 <= bottom) && (!done))

{

if (root\*2 == bottom)

maxChild = root \* 2;

else if (a[root \* 2] > a[root \* 2 + 1])

maxChild = root \* 2;

else

maxChild = root \* 2 + 1;

if (a[root] < a[maxChild])

{

temp = a[root];

a[root] = a[maxChild];

a[maxChild] = temp;

root = maxChild;

}

else

done = 1;

}

}

void heapSort(int array\_size)

{

int i, temp;

for (i = (array\_size/2)-1; i >= 0; i--)

siftDown(i, array\_size);

for (i = array\_size-1; i >= 1; i--)

{

temp = a[0];

a[0] = a[i];

a[i] = temp;

siftDown(0, i-1);

}

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int j;

cout << "Enter the elements of one-dimensional array\n";

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

{

cout << "a[" << j << "] = ";

cin >> a[j];

}

heapSort(N);

cout << "\nThe sorted elements of one-dimensional array\n";

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

cout << "a[" << j << "] = " << a[j] << "\n";

return 0;

}

Задача 45

Сортиране чрез сливане (Sort\_Merge).

#include "stdafx.h"

#include <iostream>

using namespace std;

#define N 5

int a[N];

int temp[N];

void merge(int left, int mid, int right)

{

int i, left\_end, num\_elements, tmp\_pos;

left\_end = mid - 1;

tmp\_pos = left;

num\_elements = right - left + 1;

while ((left <= left\_end) && (mid <= right))

{

if (a[left] <= a[mid])

{

temp[tmp\_pos] = a[left];

tmp\_pos = tmp\_pos + 1;

left = left +1;

}

else

{

temp[tmp\_pos] = a[mid];

tmp\_pos = tmp\_pos + 1;

mid = mid + 1;

}

}

while (left <= left\_end)

{

temp[tmp\_pos] = a[left];

left = left + 1;

tmp\_pos = tmp\_pos + 1;

}

while (mid <= right)

{

temp[tmp\_pos] = a[mid];

mid = mid + 1;

tmp\_pos = tmp\_pos + 1;

}

for (i=0; i <= num\_elements; i++)

{

a[right] = temp[right];

right = right - 1;

}

}

void m\_sort(int left, int right)

{

int mid;

if (right > left)

{

mid = (right + left) / 2;

m\_sort(left, mid);

m\_sort(mid+1, right);

merge(left, mid+1, right);

}

}

void mergeSort(int array\_size)

{

m\_sort(0, N-1);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

int j;

cout << "Enter the elements of one-dimensional array\n";

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

{

cout << "a[" << j << "] = ";

cin >> a[j];

}

mergeSort(N);

cout << "\nThe sorted elements of one-dimensional array\n";

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

cout << "a[" << j << "] = " << a[j] << "\n";

return 0;

}

Задача 46\*

Да се сравнят по бързина методите за сортиране (Задачи 39 – 45), а резултатите да се представят в табличен и графичен вид. Изследването да се проведе при вариции на входните данни – брой елементи, тип на елементите, степен на предварителна подредба и други.

#include "stdafx.h"

#include <iostream>

#include <ctime>

#define N 10000 //Брой на елементите

using namespace std;

int \_tmain(int argc, \_TCHAR\* argv[])

{

int i, a[N];

clock\_t cl;

cl = clock();

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

a[i] = rand() % 100;

cout << "Sorting ..." << "\n";

//Метод за сортиране

cl = clock() - cl;

cout << "Time: " << cl/(double)CLOCKS\_PER\_SEC << " sec.\n";

return 0;

}

XII.Списъци

Задача 47

Списък.

#include "stdafx.h"

#include <iostream>

using namespace std;

typedef int data;

typedef int keyType;

struct list {

keyType key;

data info;

struct list \*next;

};

void insertBegin(struct list \*\*L, keyType key, data x)

{

struct list \*temp;

temp = (struct list \*)malloc(sizeof(\*temp));

if(NULL == temp)

{

cout << "Nyama dostatachno pamet za nov element! \n";

return;

}

temp->next = \*L;

(\*L) = temp;

(\*L)->key = key;

(\*L)->info = x;

}

void insertAfter(struct list \*\*L, keyType key, data x)

{

struct list \*temp;

if(NULL == \*L)

{

insertBegin(L, key, x);

return;

}

temp = (struct list \*)malloc(sizeof(\*temp));

if(NULL == temp)

{

cout << "Nyama dostatachno pamet za noviya element! \n";

return;

}

temp->key = key;

temp->info = x;

temp->next = (\*L)->next;

(\*L)->next = temp;

}

void insertBefore(struct list \*\*L, keyType key, data x)

{

struct list \*temp;

if(NULL == \*L)

{

insertBegin(L, key, x);

return;

}

temp = (struct list \*)malloc(sizeof(\*temp));

if(NULL == temp)

{

cout << "Nyama dostatachno pamet za noviya element! \n";

return;

}

\*temp = \*\*L;

(\*L)->next = temp;

(\*L)->key = key;

(\*L)->info = x;

}

void deleteNode(struct list \*\*L, keyType key)

{

struct list \*current = \*L;

struct list \*save;

if((\*L)->key == key)

{

current = (\*L)->next;

free(\*L);

(\*L) = current;

return;

}

while(current->next != NULL && current->next->key != key)

{

current = current->next;

}

if(NULL == current->next)

{

cout << "Greshka: Elementat za iztrivane ne e nameren!\n";

return;

}

else

{

save = current->next;

current->next = current->next->next;

free(save);

}

}

void print(struct list \*L)

{

while(NULL != L)

{

cout << L->key << "(" << L->info << ") ";

L = L->next;

}cout << "\n";

}

struct list\* search(struct list \*L, keyType key)

{

while(L != NULL)

{

if(L->key == key)

return L;

L = L->next;

}

return NULL;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

struct list \*L = NULL;

int i, edata;

insertBegin(&L, 0, 42);

for(i=1; i<6; i++)

{

edata = rand()%100;

cout << "Vmakvane predi: " << i << "(" << edata << ")" << "\n";

insertBefore(&L, i, edata);

}

for(i=6; i<10; i++)

{

edata = rand()%100;

cout << "Vmakvane sled: " << i << "(" << edata << ")" << "\n";

insertAfter(&L, i, edata);

}

print(L);

deleteNode(&L, 9);

print(L);

deleteNode(&L, 0);

print(L);

deleteNode(&L, 3);

print(L);

deleteNode(&L, 5);

print(L);

deleteNode(&L, 5);

return 0;

}

Задача 48\*

Съставете алгоритъм и напишете програма, проверяваща за симетричност даден низ, въведен чрез главната функция. Другото наименование на симетричен низ е палиндром.

Задачи за изпълнение

1. Изпълнете, разгледайте и анализирайте решените задачи – номера 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45 и 47.

2. Съставете алгоритъм и напишете програмен код на нерешените задачи – номера 33\*, 46\* и 48\*