МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра АПУ

ОТЧЕТ

по лабораторной работе № 6

по дисциплине «Алгоритмы и структуры данных»

Тема: Эвристические алгоритмы

Студентка гр. 1361	_____	Горбунова Д. А.
Студентка гр. 1361	_____	Токарева У. В.
Преподаватель	_____	Беляев А. В.

Санкт-Петербург
2022

Цель работы: ознакомление с принципами работы эвристических алгоритмов при решении NP-сложных задач.

Теоретическая часть

Эвристический алгоритм — это алгоритм, предназначенный для решения проблемы значительно более быстрым и эффективным способом, чем традиционные методы, за счет жертвования оптимальностью, точностью или полнотой ради скорости.

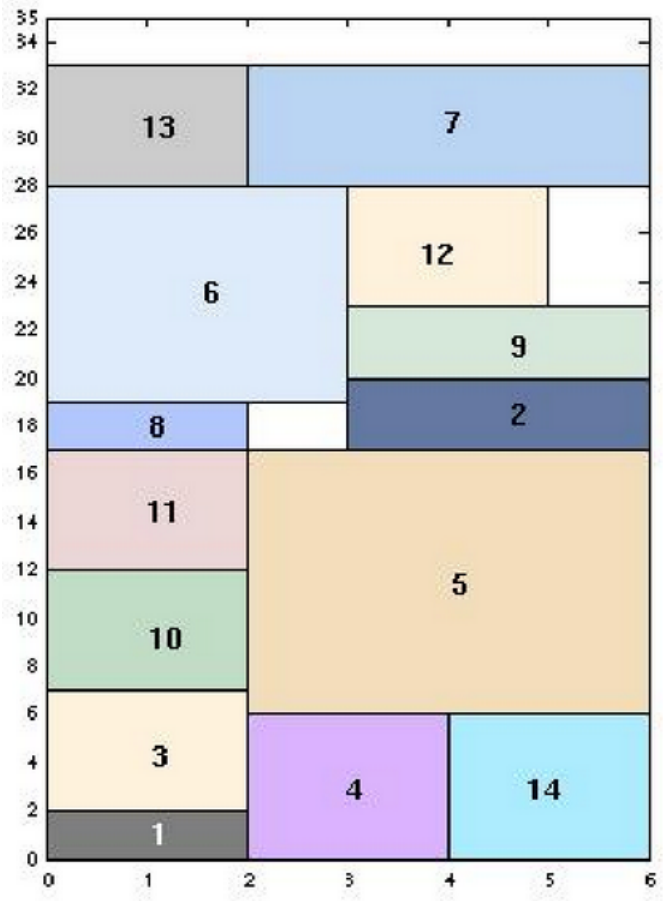
Эвристические алгоритмы часто используются для решения NP-полных задач. В этих задачах не существует известного эффективного способа быстрого и точного нахождения решения; однако, при этом если решение получено, то его можно проверить, в т.ч. в некоторых случаях оценить его расхождение от оптимального.

В зависимости от задачи эвристические алгоритмы могут быть использованы для получения итогового решения, либо же использоваться для построения некоторого базового решения, которое в дальнейшем будет улучшено другими алгоритмами оптимизации.

Пример задачи

Рассмотрим работу эвристических алгоритмов на примере NP-сложной задачи, встречающейся во многих производственных сферах: оптимальная упаковка (или раскрой) прямоугольных объектов в двумерном пространстве.

Дан набор двумерных прямоугольных объектов, заданных своей шириной (в дальнейшем – координата X) и высотой (координата Y). Для упрощения задачи условимся, что поворачивать объекты (менять между собой координаты X и Y) нельзя. Также дана лента фиксированной ширины, но неограниченная, с одной стороны, по высоте (например, вверх). Требуется найти оптимальное расположение всех объектов данного набора на данной ленте без наложений таким образом, чтобы итоговая высота, занимаемая объектами на ленте, была минимальна.



(пример упаковки набора из 14 объектов, ширина ленты = 6, итоговая высота = 33) [1]

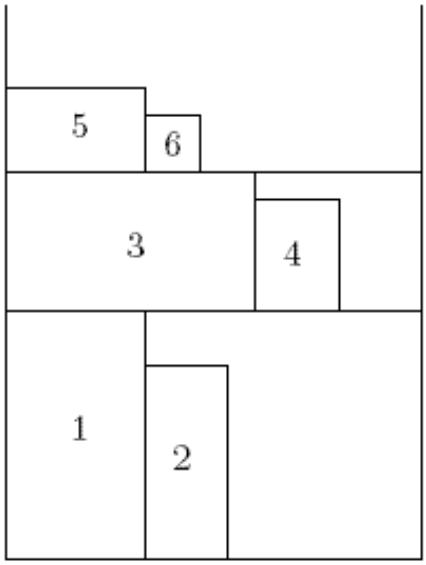
Задача поиска оптимального решения является вычислительно крайне сложной, поэтому на практике применяются различные эвристические алгоритмы. Рассмотрим наиболее простые из них.

Общий принцип

Все рассматриваемые ниже алгоритмы пытаются разбить ленту на ряды (различной высоты, постепенно убывающей снизу вверх) и размещать объекты слева направо в рядах. Перед обработкой все объекты сортируются по невозрастанию высоты (этим обеспечивается гарантия того, что очередной объект будет заведомо меньшим по высоте чем объект, который начал ряд).

Next-Fit Decreasing Height

Алгоритм пытается разместить объект в самый верхний ряд (в данном алгоритме он считается единственным доступным для заполнения). Когда очередной объект не удается разместить в этот ряду (из-за своей ширины), ряд считается закрытым и открывается новый ряд, высотой равный добавляемому объекту, который становится самым левым в нем.

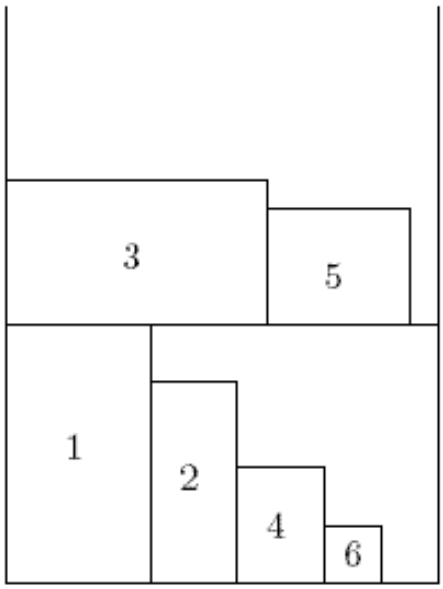


NFDH

Обратите внимание: на иллюстрации объект «4» вынужден быть размещен во втором ряду, хотя в первом (самом нижнем) ему было бы достаточно места, поскольку самый нижний ряд считается уже «закрытым». [2]

First-Fit Decreasing Height

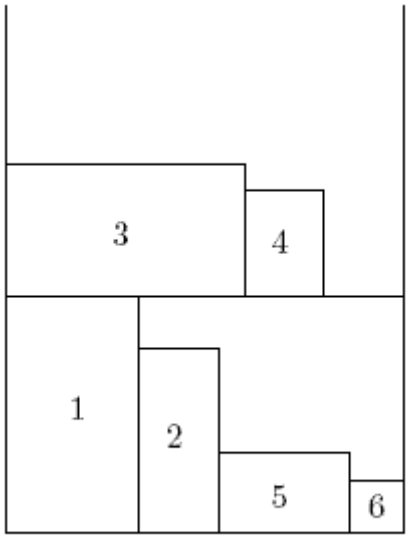
В отличие от предыдущего алгоритма все ряды считаются доступными для дополнения новыми объектами. Алгоритм пытается разместить очередной объект в первый снизу ряд, в котором объекту окажется достаточно места по ширине. Только если ни одного такого ряда не найдено, объект создает новый ряд.



FFDH

Best-Fit Decreasing Height

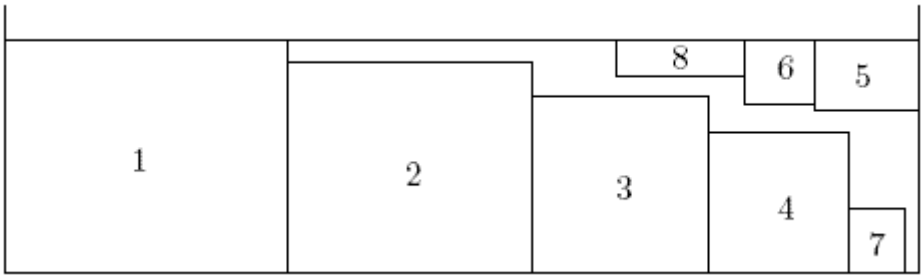
Аналогично предыдущему алгоритму все ряды считаются доступными для дополнения новыми объектами. Алгоритм пытается разместить очередной объект в ряд, в котором объекту окажется достаточно места по ширине; при этом если таких рядов несколько, то будет выбран ряд, наиболее заполненный по ширине на текущий момент.



BFDH

Обратите внимание: на иллюстрации объект «4» был размещен на втором ряду, т.к. край объекта «3» находился правее края объекта «2», что в итоге привело к более плотному заполнению самого нижнего ряда, чем в предыдущем алгоритме.

Существует множество модификаций описанных алгоритмов, а также алгоритмы, которые используют другие методы размещения объектов, например, вот так:



Почти каждый из эвристических алгоритмов имеет свои преимущества и недостатки, и выбор конкретного алгоритма существенно зависит от статистических характеристик объектов, а также от желаемого отношения «скорость/качество» поиска решения.

Исходный код программы Next-Fit Decreasing Height

```
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <vector>
using namespace std;
 struct Rectangle{
     int length;
     int width;
     int flag; };
 struct Box{
     int length=0;
     int width = 1000;
     int count_subject=0;
     int bloks[100]; };
void PrintFile(int flag, Rectangle * rec){
    int count=0;
    while (count < flag ){
        cout << "Rectangle: "<< count << " \tLength: "<<  rec[count].length;
        cout << " \tWidth: "<< rec[count].width<< endl;
        count++;
    }
```

```
    return;
}

void Sort_Rec (Rectangle * rec, int flag){
    for (int i=0; i<flag; i++)
        for (int j=0; j<flag; j++)
            if (rec[i].length >= rec[j].length)
                swap(rec[i], rec[j]);
}
void NFDH(Rectangle * rec, Box box, int flag){
    int level=0;
    for (int i=0; i<flag-1; i++){
        if (rec[i].length !=0){
            box.length += rec[i].length;
            level ++;
            int curr_width=0;
            cout << "Level: "<< level<< " Length: "<< rec[i].length<< " Width: ";
            for (int j=i; j<flag;j++){
                if ((box.width-(rec[j].width+curr_width)>=0)){
                    curr_width +=rec[j].width;
                    cout << rec[j].width<<",";
                    rec[j].length =0;
```

```
                    rec[j].width =0; }
                else break; }
        cout <<"\nResidual_width: " << box.width - curr_width <<endl<< endl;}}}
int main(){
    FILE *file;
    int scan_error;
    int length, width, flag=133;
    Rectangle rectangle[flag];
    Box box1;
    file = fopen ("C:\\Users\\Leera\\CLionProjects\\A_CD_LR6\\ads_lab6_rnd.txt", "r");
    if (file == nullptr){
        cout << "Error open file\n";
        return 1;
    }
    int count=0;
    do {
        scan_error = fscanf(file, "%d%d", &length,&width);
        rectangle[count].length = length;
        rectangle[count].width = width;
        count ++;
    }while (scan_error!=EOF || count < flag);
    Sort_Rec(rectangle,flag);
```

```
    PrintFile(flag, rectangle);
    cout << "\n----------------------------------------------------------------\n\n\n";
   cout << "\t\t\tNext-Fit Decreasing Height: \n\n\n";
    NFDH(rectangle,box1, flag);
    cout << "----------------------------------------------------------------\n\n Summary length: "<< box1.length<< endl;
    return 0;}
```

Исходный код программы First-Fit Decreasing Height

```
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <vector>
using namespace std;
 struct Rectangle{
     int length;
     int width;
     int flag;};
 struct Box{
     int length=0;
     int width = 1000;
     int count_subject=0;
```

```
     int bloks[100];};
void PrintFile(int flag, Rectangle * rec){
    int count=0;
    while (count < flag ){
        cout << "Rectangle: "<< count << " \tLength: "<<  rec[count].length;
        cout << " \tWidth: "<< rec[count].width<< endl;
        count++;}
    return;}
void Sort_Rec (Rectangle * rec, int flag){
    for (int i=0; i<flag; i++)
        for (int j=0; j<flag; j++)
            if (rec[i].length >= rec[j].length)
                swap(rec[i], rec[j]);}
void FFDH(Rectangle * rec, Box* box, int flag){
    int level=0;
    for (int i=0; i< flag; i++){
        int cnt =1;
        while (cnt){
            if (box[level].width >= rec[i].width){
                box[level].width -= rec[i].width;
                box [level].bloks[box[level].count_subject] = rec[i].width;
                box[level].count_subject++;
```

```
                if (rec[i].length > box[level].length)
                    box[level].length = rec [i].length;
                level=0;
                cnt=0;}
            else level++;}}
    int length=0;
    int k=1;
    for (int i=0; i<9;i++){
        length += box[i].length;
        cout << "Level: "<< k << " \t";
        cout << " Length: "<< box[i].length << " Width: ";
        for (int  j=0 ; j<box[i].count_subject; j++)
            cout << box[i].bloks[j]<< " ";
        cout << "\nResidual_width: "<< box[i].width<< "\n\n";
        k++;}
    cout << "\n----------------------------------------------------------------\n\nSummary length:" <<length<<" \n";}
int main(){
    FILE *file;
    int scan_error;
    int length, width, flag=133;
    Rectangle rectangle[flag];
    Box box1,box2 [10];
```

```
    file = fopen ("C:\\Users\\Leera\\CLionProjects\\A_CD_LR6\\ads_lab6_rnd.txt", "r");
    if (file == nullptr){
        cout << "Error open file\n";
        return 1;
    }
    int count=0;
    do {
        scan_error = fscanf(file, "%d%d", &length,&width);
        rectangle[count].length = length;
        rectangle[count].width = width;
        count ++;
    }while (scan_error!=EOF || count < flag); //заполнение массива структур прямоугольника
    Sort_Rec(rectangle,flag);
    PrintFile(flag, rectangle);
    cout << "\n----------------------------------------------------------------\n\n\n";
    cout << "\t\t\tFirst-Fit Decreasing Height: \n\n\n";
    FFDH(rectangle,box2, flag);


    return 0;
}
```

Результат работы программы.

```
                      Next-Fit Decreasing Height:
Level: 1 Length: 475 Width: 193,194,528,85,
Residual_width: 0

Level: 2 Length: 272 Width: 194,130,69,191,60,14,105,133,
Residual_width: 104

Level: 3 Length: 148 Width: 151,105,15,12,277,76,37,74,122,6,85,
Residual_width: 40

Level: 4 Length: 114 Width: 180,124,21,37,196,139,8,144,18,100,2,
Residual_width: 31

Level: 5 Length: 88 Width: 101,45,122,55,53,6,194,55,95,72,118,
Residual_width: 84

Level: 6 Length: 67 Width: 124,106,152,194,90,167,19,
Residual_width: 148

Level: 7 Length: 44 Width: 217,133,24,131,35,5,1,12,12,86,110,30,6,32,21,14,12,24,5,18,18,15,32,6,
Residual_width: 1

Level: 8 Length: 15 Width: 15,2,8,15,30,19,15,40,82,26,1,82,33,25,106,14,13,1,2,4,37,133,9,18,53,19,133,5,6,4,38,
Residual_width: 12

Level: 9 Length: 6 Width: 49,8,65,9,5,43,4,19,6,21,4,3,8,19,4,6,6,11,2,92,88,12,24,4,122,9,
Residual_width: 357

------------------------------------------------------------
 Summary length: 1229
```

Рисунок 1 – Вывод алгоритма NFDH

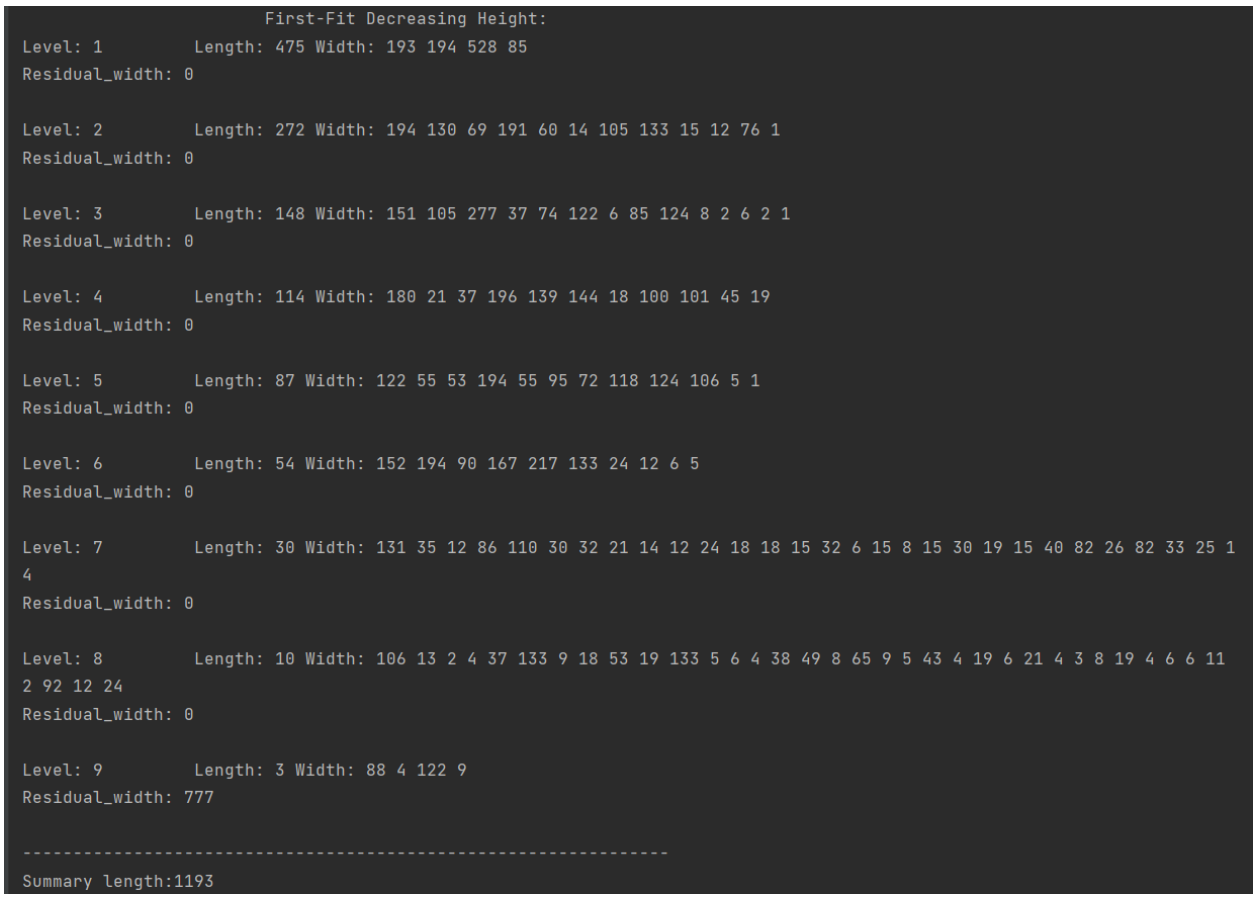


Рисунок 2 – Вывод алгоритма FFDH

ВЫВОД

В ходе выполнения данной лабораторной работы были изучены алгоритмы NFDH и FFDH. Алгоритм FFDH более эффективен чем NFDH, однако оба алгоритма приближены к эталонному.