CHUYÊN ĐỀ: THAM LAM

1. Giới thiệu

Phương pháp tham lam là một mô hình thuật toán dùng để giải quyết ***bài toán tối ưu tổ hợp***. Phương pháp này xây dựng lời giải của bài toán qua **từng bước**, tại mỗi bước luôn chọn kết quả **tối ưu nhất** cho bước đó.

Ưu điểm: Có thể áp dụng cho bài toán tối ưu, ý tưởng đơn giản, dễ cài đặt.

Hạn chế: thuật toán tham lam không phải khi nào cũng cho kết quả tối ưu.

2. Một số ví dụ

2.1. Ghép cặp số:

Hãy ghép ***2n*** số thành ***n*** cặp số sao cho tổng của tích các cặp số là lớn nhất biết rằng mỗi số chỉ được chọn ghép một lần.

**Dữ liệu:** ***n*** và ***a[1], a[2],..,a[2n]*** nhập từ bàn phím.

**Kết quả:** ghi ra màn hình tổng và các cặp số.

Ví dụ:

|  |  |
| --- | --- |
| **Dữ liệu** | **Kết quả** |
| 2  1 3 4 2 | - Tổng: 14  - Các cặp: (1,2); (3,4) |
| 5  1 3 -4 2 0 1 | - Tổng: 7  - Các cặp: (-4,0), (1,1); (3,2) |

**2.1.1. Phân tích bài toán**

Input: N, a[1],a[2],..,a[2n]

Output: tổng S, N cặp số

**2.1.2. Ý tưởng thuật toán**

Tại mỗi bước chọn cặp (a,b) sao cho a\*b lớn nhất có thể.

**2.1.3. Thuật toán**

1. Nhập N, a[1],..,a[2n] từ bàn phím.

2. Sắp xếp các a[i] giảm dần.

3. Tham

Tong:=0

For i=1 to n do Tong:=Tong+a[2i-1]\*a[2i]

4. In kết quả:

Tổng lớn nhất là: Tong

Cách ghép các số là: (a[2i-1],a[2i]) với i=1…n.

#include <bits/stdc++.h>

using namespace std;

int main()

{

    int n, i;

    freopen("in.inp", "r", stdin);

    freopen("out.out", "w", stdout);

    cin >> n;

    int a[2 \* n];

    for (int i = 1; i <= 2 \* n; i++)

    {

        cin >> a[i];

    }

    sort(a, a + n \* 2 + 1, greater<int>());

    int s = 0;

    i = 1;

    while (i < 2 \* n)

    {

        s = s + a[i] \* a[i + 1];

        cout << a[i] << " " << a[i + 1] << endl;

        i = i + 2;

    }

    cout << s;

    return 0;

}

2.2. Mua kẹo:

Siêu thị bày bán ***n*** gói kẹo, gói thứ ***i*** có giá là ***a[i] (1<=i<=n)***. Tí muốn dùng số tiền ***m*** để mua kẹo. Hãy giúp Tí chọn các gói để mua sao cho số gói kẹo mua được nhiều nhất. Nếu cùng mua được số gói như nhau thì chọn phương án tốn ít nhất.

**Dữ liệu:** file văn bản **MUAKEO.INP**

- Dòng đầu ghi 2 số nguyên ***n*** và ***m***

- Dòng thứ hai ghi ***n*** số nguyên ***a[1], a[2],..,a[n].***

**Kết quả:** file văn bản **MUAKEO.OUT**

- Dòng đầu ghi số gói mua được

- Dòng thứ 2: ghi số tiền còn lại sau khi mua.

*\* Các số trong file cách nhau một dâu cách*

Ví dụ:

|  |  |
| --- | --- |
| **MUAKEO.INP** | **MUAKEO.OUT** |
| 3 5  3 4 2 | 2  0 |
| 5 7  1 3 7 1 4 | 3  2 |

**2.2.1. Phân tích bài toán**

input: n, m, a[1],a[2],..,a[n]

Output: số gói S và số tiền còn lại T

**2.2.2. Ý tưởng thuật toán**

Khi mua một món quà ta chọn món quà ít tiền nhất có thể.

**2.2.3. Thuật toán**

1. Đọc N,M, a[1],..,a[n] từ file

2. Sắp xếp các a[i] tăng dần

3. Tham

S:=0; T:=M;

For i:=1 to n do If T>=a[i] then

S:=S+1;

T:=T-a[i]

4. In kết quả

Ghi vào file trên mỗi dòng lần lượt S và T.

#include <bits/stdc++.h>

using namespace std;

int main()

{

    int n, i, m;

    freopen("in.inp", "r", stdin);

    freopen("out.out", "w", stdout);

    cin >> n;

    cin >> m;

    int a[n];

    for (int i = 0; i < n; i++)

    {

        cin >> a[i];

    }

    sort(a, a + n);

    int s = 0;

    int t = m;

    for (int i = 0; i < n; i++)

    {

        if (a[i] <= t)

        {

            s++;

            t = t - a[i];

        }

        else

        {

            break;

        }

    }

    cout << s << endl

         << t;

    return 0;

}

3. Xếp lịch

Trong dịp Tết nhà hát trung tâm thành phố nhận được***n*** yêu cầu tổ chức hoạt động. Hoạt động thứ ***i*** bắt đầu từ ***a[i]*** và kết thúc ngay trước ***b[i]*** *(a[i]<b[i])*. Nhà hát tại một thời điểm chỉ tổ chức được tối đa một hoạt động. Hãy xác định xem số lượng tối đa các hoạt động mà nhà hát có thể tổ chức được và chỉ rõ các hoạt động được chọn.

**Dữ liệu:** file văn bản **XEPLICH.INP**

- Dòng đầu ghi số nguyên ***n*** 

- Dòng thứ hai ghi ***n*** số nguyên ***a[1], a[2],..,a[n]***

- Dòng thứ ba ghi  ***n*** số nguyên ***b[1], b[2],..,b[n]***

**Kết quả:** file văn bản **XEPLICH.OUT**

- Ghi một số duy nhất là số hoạt động tối đa phục vụ được.

*\* Các số trong file cách nhau một dâu cách*

Ví dụ:

|  |  |
| --- | --- |
| **XEPLICH.INP** | **XEPLICH.OUT** |
| 11  1 0 3 6 8 12 2 8 5 5 3  4 6 8 10 12 14 13 11 9 7 5 | 4  1 10 8 6 |

**3.1. Phân tích bài toán**

DLV: N, a[1],..,a[n] và b[1],..,b[n]

KQ: Số hoạt động tối đa chọn được S

**3.2. Ý tưởng thuật toán**

i. Mỗi lần chọn một hoạt động bắt đầu thời điểm t (banđầu t=0)

ii. Khi chọn ta chọn hoạt động kết thúc sớm nhất

#include <bits/stdc++.h>

using namespace std;

struct hd

{

    int start;

    int finish;

    int id;

};

bool cmp(hd *a*, hd *b*)

{

    return a.finish < b.finish;

}

int main()

{

    int n, i;

    freopen("in.inp", "r", stdin);

    freopen("out.out", "w", stdout);

    cin >> n;

    hd a[n];

    for (int i = 1; i <= n; i++)

    {

        cin >> a[i].start;

        a[i].id = i;

    }

    for (int i = 1; i <= n; i++)

    {

        cin >> a[i].finish;

    }

    sort(a + 1, a + n + 1, cmp);

    int d = 1;

    int j = 1;

    vector<int> res;

    res.push\_back(a[1].id);

    for (int i = 2; i <= n; i++)

    {

        if (a[j].finish <= a[i].start)

        {

            d++;

            res.push\_back(a[i].id);

            j = i;

        }

    }

    cout << d << endl;

    for (int x : res)

    {

        cout << x << " ";

    }

    return 0;

}

4. Cho thuê xe

Công ty lữ hành Alpha nhận được***N*** yêu cầu thuê xe trong cùng một thời điểm, yêu cầu thứ ***i*** đi đoạn đường dài ***a[i]*** kilomet ***(i=1..N)***. Hiện tại công ty có ***M*** chiếc xe có thể phục vụ ***(N<=M)***, xe thứ ***j*** đi một kilomet chi phí ***b[j] (j=1..M)***. Để thu được lợi nhuận cao nhất công ty cần bố trí các xe phục vụ sao cho tổng chi phí cho các chiếc xe là nhỏ nhất biết rằng mỗi xe chỉ phục vụ một yêu cầu.

Hãy xác định giá trị chi phí nhỏ nhất đó là bao nhiêu.

**Dữ liệu:** file văn bản **CHTHUEXE.INP**

- Dòng đầu ghi số nguyên ***N*** và ***M***

- Dòng thứ hai ghi ***n*** số nguyên ***a[1], a[2],..,a[N]***

- Dòng thứ ba ghi ***m*** số nguyên ***b[1], b[2],..,b[M]***

**Kết quả:** file văn bản **CHTHUEXE.OUT**

- Ghi một số duy nhất chi phí tối thiểu.

- Dòng thứ i: ghi chỉ số xe phục vụ đoàn

*\* Các số trong file cách nhau một dâu cách*

Ví dụ:

|  |  |
| --- | --- |
| **CHTHUEXE.INP** | **CHTHUEXE.OUT** |
| 4 5  10 15 18 10  5 5 10 6 10 | 325  3 2 1 4 |

**Giới hạn:**

***1<=N,M<=200; 1<=a[i]<=200; 1<=b[j]<=100;***

#include <bits/stdc++.h>

using namespace std;

int main()

{

    int n, m;

    freopen("in.inp", "r", stdin);

    freopen("out.out", "w", stdout);

    cin >> n >> m;

    vector<pair<int, int>> a(n);

    for (int i = 0; i < n; i++)

    {

        cin >> a[i].first;

        a[i].second = i + 1;

    }

    vector<pair<int, int>> b(m);

    for (int i = 0; i < m; i++)

    {

        cin >> b[i].first;

        b[i].second = i + 1;

    }

    sort(a.begin(), a.end(), greater<pair<int, int>>());

    sort(b.begin(), b.end());

    int s = 0;

    vector<int> res(n);

    for (int i = 0; i < n; i++)

    {

        s += a[i].first \* b[i].first;

        res[a[i].second - 1] = b[i].second;

    }

    cout << s << endl;

    for (int x : res)

    {

        cout << x << " ";

    }

    return 0;

}

5. Xếp phòng thi (Room.pas)

Trong một lần tổ chức Olympic Tin học sinh viên, có N cuộc thi được đánh số từ 1 đến N. Cuộc thi thứ I có thời điểm bắt đầu là Si và kết thúc là Fi. Tại mỗi thời điểm trong mỗi phòng thi có không quá một cuộc thi diễn ra ngoại trừ trường hợp thời điểm kết thúc một cuộc thi có thể đồng thời là thời điểm bắt đầu của một cuộc thi khác.

Hãy xếp phòng thi cho tất cả các cuộc thi sao cho số phòng cần sử dụng là ít nhất.

Dữ liệu vào: Từ file văn bản Room.inp gồm:

- Dòng thứ nhất ghi số nguyên dương N (n<=1000)

- Trên dòng thứ i trong N dòng tiếp theo là hai số nguyên Si và Fi .

Dữ liệu ra:

- Ghi kết quả vào tệp Room.out là số phòng cần sử dụng.

- chỉ rõ vị trí các phòng

|  |  |
| --- | --- |
| Room.inp | Room.out |
| 5  0 2  1 2  3 4  2 5  4 5 | 2  1 3 |

#include <bits/stdc++.h>

using namespace std;

struct hd

{

    int start;

    int finish;

    int id;

};

bool cmp(hd *a*, hd *b*)

{

    return *a*.finish < *b*.finish;

}

int main()

{

    int n, i;

    freopen("in.inp", "r", stdin);

    freopen("out.out", "w", stdout);

    cin >> n;

    hd a[n];

    for (int i = 1; i <= n; i++)

    {

        cin >> a[i].start >> a[i].finish;

        a[i].id = i;

    }

    sort(a + 1, a + n + 1, cmp);

    int d = 1;

    int j = 1;

    vector<int> res;

    res.push\_back(a[1].id);

    for (int i = 2; i <= n; i++)

    {

        if (a[j].finish < a[i].start)

        {

            d++;

            res.push\_back(a[i].id);

            j = i;

        }

    }

    cout << d << endl;

    for (int x : res)

    {

        cout << x << " ";

    }

    return 0;

}

**6.** Cửa hàng bán elẻ xáng dầu PETROL-HUE thường nhập hang từ công ty mẹ ở khu chế xuất Linh Trung - Thành phố Hồ Chí Minh.

Trong mỗi đợt nhập hàng, tùy thuộc vào lượng xăng còn tồn kho mà cửa hàng sẽ quyết định nhập thêm số mét khối (m3) xăng để dự trữ bán cho khách phù hợp với thể tích các bồn chứa hiện có cửa hàng.

Giả sử cửa hàng có n loại bồn chứa xăng thuộc các dạng vuông, trụ đứng và trụ nằm mỗi loại có thể tích ạ[i] (m3), cửa hàng cần nhập thêm k (m3) xăng.

Yêu cầu: Hãy cho biết cửa hàng có thể chứa lượng xăng cần nhập vào những loại bồn chứa xăng nào để số bồn chứa xăng cần sử dụng là ít nhất

Dữ liệu vào: Cho bởi file PETROL.INP gồm:

- Dòng thứ nhất chứa hai số n, k nguyên dương lần lượt là số loại bồn chứa xăng và sô lượng xăng cần nhập (1 < n < 1000; 1 < k < 100000).

- Dòng thứ hai chứa n số nguyên duơng a[l], a[2] ••• a[n] lần lượt là thể tích(của loại bồn chứa thứ i (1 < a[i] < 1000)

Kết quá: Ghi ra file PETROL.OUT duy nhất một giá trị cho biết số bồn chứa xăng ít nhất cần sử dụng để chứa lượng xăng cần nhập.

**PETROL.INP**

3 15

2 3 4

PETROL.OUT

4

#include <bits/stdc++.h>

using namespace std;

int main()

{

    freopen("in.inp", "r", stdin);

    freopen("out.out", "w", stdout);

    int n, k;

    cin >> n >> k;

    int a[n];

    for (int i = 0; i < n; i++)

    {

        cin >> a[i];

    }

    sort(a, a + n, greater<int>());

    int d = 0;

    int s = k;

    while (s > 0)

    {

        for (int i = 0; i < n; i++)

        {

            if (s >= a[i])

            {

                d++;

                s -= a[i];

                break;

            }

        }

    }

    cout << d;

    return 0;

}

**7. Cho 1 tập công việc, mỗi công việc có 1 deadline và 1 giá trị lợi nhuận nếu hoàn thành trước deadline. Hãy sắp xếp các công việc để tối đa hóa tổng lợi nhuận.**

**INP**

**Cviec deadline lợi nhuận**

**5**

**1 2 100**

**2 1 19**

**3 2 27**

**4 1 25**

**5 3 15**

**OUT**

**3 1 5**

**HD: Sắp xếp công việc theo lợi nhuận giảm dần và chọn các công việc từ danh sách này, cố gắng hoàn thành công việc trước deadline của nó.**

#include <bits/stdc++.h>

using namespace std;

struct Job

{

    int id, dl, ln;

};

Job a[100];

bool cmp(Job *x*, Job *y*)

{

    return (x.ln > y.ln);

}

int main()

{

    freopen("in.inp", "r", stdin);

    freopen("out.out", "w", stdout);

    int id, dl, ln, n;

    int i;

    cin >> n;

    for (i = 0; i < n; i++)

    {

        cin >> a[i].id >> a[i].dl >> a[i].ln;

    }

    sort(a, a + n, cmp);

    int s = 0;

    int kq[100] = {0};

    bool check[100] = {false};

    for (i = 0; i < n; i++)

    {

        for (int j = min(n, a[i].dl); j >= 1; j--)

        {

            if (check[j] == false)

            {

                check[j] = true;

                kq[j] = i;

                s += a[i].ln;

                break;

            }

        }

    }

    cout << s << endl;

    for (i = 0; i < n; i++)

    {

        if (check[i] == true)

        {

            cout << a[kq[i]].id << " ";

        }

    }

    return 0;

}