Thuật toán ứng dụng

Giảng viên : Đỗ Tuấn Anh, Lê Quốc Trung
TA: Trần Thanh Tùng

Viện Công Nghệ Thông Tin và Truyền Thông
Trường Đại học Bách Khoa Hà Nội

Tháng 3, năm 2020

Mục lục

1 TAXI

2 KNAPSAC

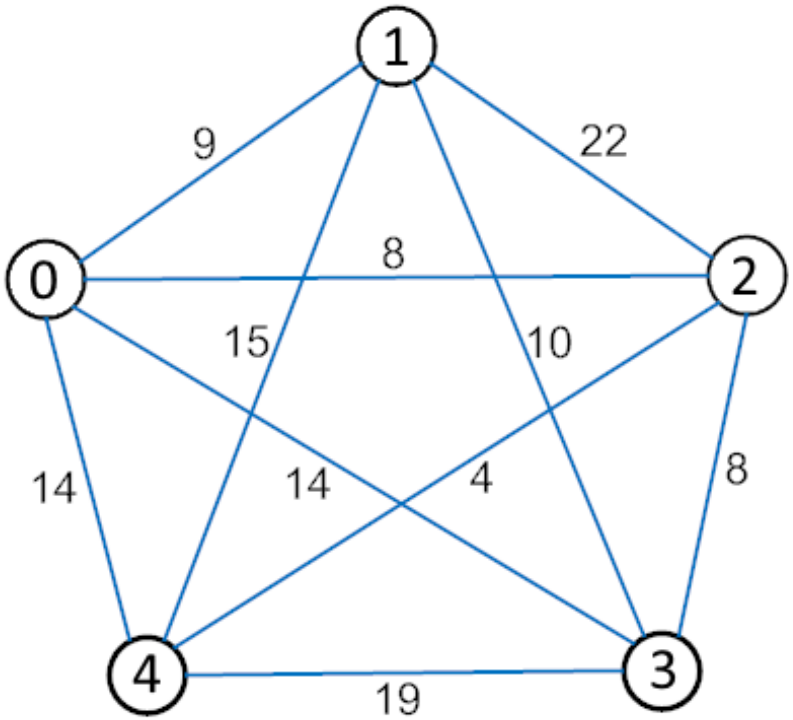
3 BCA

Lịch sử

- Nêu ra lần đầu tiên năm 1930 về tối ưu hóa
- Dưới dạng bài toán “The Saleman Problem”

Phát biểu bài toán

- Một tài xế Taxi xuất phát từ điểm 0, và nếu khoảng cách giữa hai điểm bất kỳ được biết thì đâu là đường đi ngắn nhất mà người Taxi có thể thực hiện sao cho đi hết tất cả các điểm mỗi điểm một lần để quay về lại điểm A.
- Đầu vào : khoảng cách giữa các điểm, tài xế Taxi xuất phát từ điểm 0, và có n điểm từ $1, 2, 3, ..n$ cần đi qua.
- Đầu ra : đường đi ngắn nhất $0 \rightarrow i \rightarrow j \ldots \rightarrow 0$
- Dưới dạng đồ thị : bài toán người lái taxi được mô hình hóa như một đồ thị vô hướng có trọng số, trong đó mỗi điểm đến là một đỉnh của đồ thị, đường đi từ một điểm đến điểm khác là khoảng cách hay chính là độ dài cạnh.



- Tổng quãng đường đi từ
  $0 \to 2 \to 4 \to 1 \to 3 \to 0 = 8 + 4 + 15 + 10 + 4 = 51$
- Tổng quãng đường đi từ
  $0 \to 1 \to 4 \to 2 \to 3 \to 0 = 9 + 15 + 4 + 8 + 4 = 40$

Ứng dụng

- Lập kế hoạch như bài toán Taxi là tối ưu trong quãng đường phục vụ, bài toán Người bán hàng,...
- Thiết kế vi mạch $\rightarrow$ Tối ưu về đường nối, điểm hàn
- Trong lĩnh vực phân tích gen sinh học
- Trong lĩnh vực du lịch

TAXI

- Có n hành khách được đánh số từ 1 tới n.
- Có $2n + 1$ địa điểm được đánh số từ 0 tới $2n$
- Hành khách thứ i muốn đi từ địa điểm thứ i đến địa điểm thứ $i + n$
- Taxi xuất phát ở địa điểm thứ 0 và phải phục vụ n hành khách và quay lại địa điểm thứ 0 sao cho không điểm nào được đi lại 2 lần và tại một thời điểm chỉ có 1 hành khách được phục vụ.
- Cho khoảng cách giữa các địa điểm. Tính quãng đường nhỏ nhất mà tài xế Taxi phải đi.

Thuật toán

- Nhận xét với mỗi cách chọn đường đi ta sẽ ánh xạ về được một hoán vị từ 1 đến n.
- Ta duyệt hết $n!$ hoán vị.
- Với mỗi hoán vị tính toán khoảng cách phải di chuyển.
- Độ phức tạp thuật toán $O(n! * n)$, có thể cải tiến xuống $O(n!)$ hoặc thậm chí $O(2^n * n^2)$.

Công thức

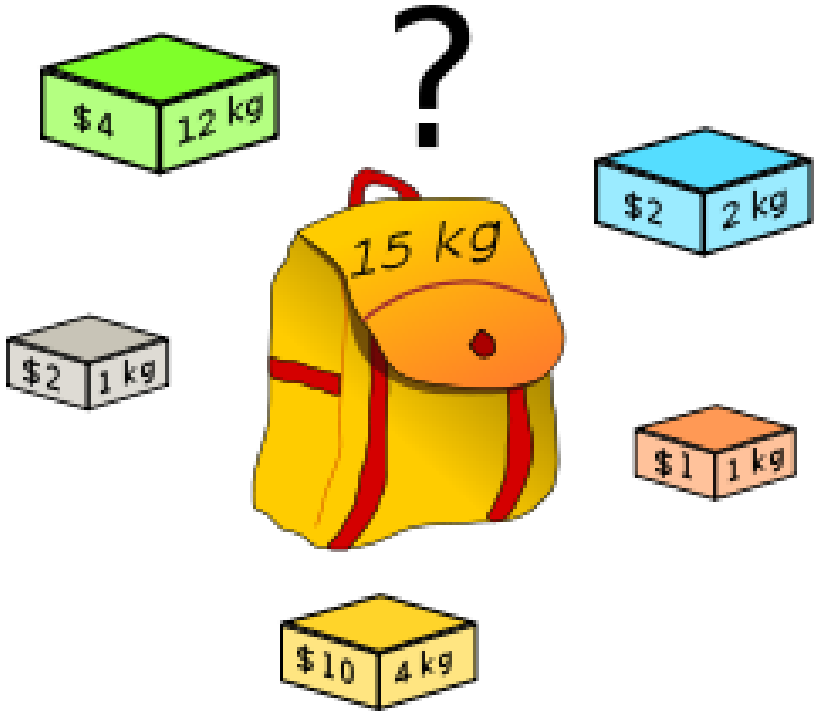
- Gọi $c[i][j]$ là khoảng cách di chuyển từ địa điểm i đến địa điểm j
- Gọi một hoán vị có dạng $a_1, a_2, ..., a_n$
- Công thức : $\sum_{i=1}^{n} c[a_i][a_i + n] + \sum_{i=1}^{n-1} c[a_i + n][a_{i+1}] + c[0][a_1] + c[a_n + n][0]$

Code

```
1
2  int calc(int a[]) {
3      // tra ve chi phi di chuyen cua hoan vi a
4      // tinh theo cong thuc da neu
5  }
6
7
8  int main() {
9      Nhap n
10     Nhap ma tran c[i][j]
11     Khoi tao hoan vi a[] = {1, 2, ..., n}
12     answer = INF
13     while (1) {
14         cost = calc(a)
15         if (a == {n, n - 1, ..., 1}) {
16             break;
17         }
18         a = next_permutation(a)
19     }
20     In ra answer
21 }
```

KNAPSAC

- Một nhà thám hiểm cần đem theo một cái túi có trọng lượng không quá m.
- Có n đồ vật có thể đem theo.
- Đồ vật thứ i có trọng lượng a_i và giá trị sử dụng c_i.
- Hỏi nhà thám hiểm cần đem theo những đồ vật nào để cho tổng giá trị sử dụng là lớn nhất mà tổng trọng lượng đồ vật mang theo cái túi không vượt quá m?
- Ghi ra duy nhất một số là tổng giá trị lớn nhất tìm được của các đồ vật cho vào túi.


?
$4
12 kg
$2
2 kg
15 kg
$2
1 kg
$1
1 kg
$10
4 kg

Ứng dụng

- Xếp đồ đi du lịch
- Xếp hàng trong lĩnh vực vận chuyển hàng hóa

Thuật toán 1

- Mỗi cách chọn lấy các đồ vật tương ứng với một dãy nhị phân độ dài n. bit thứ i là 0/1 tương ứng là không lấy/có lấy đồ vật thứ i.
- Duyệt hết các xâu nhị phân độ dài n và tìm nghiệm tốt nhất.
- Để xét hết các xâu nhị phân độ dài n, có thể dùng đệ quy - quay lui hoặc chuyển đổi giữa thập phân với nhị phân.
- Độ phức tạp $O(2^n \times n)$

Code 1

```
22  main(){
23      Nhap n, m
24      Nhap mang a, c
25      answer = 0;
26      for (int mask = 1 << n; mask--; ) { // mask = 0..2^n
27          int sum_weight = 0, sum_value = 0;
28          for (int i = 0; i < n; ++i)
29              if (mask >> i & 1){ // bit thu i = 1
30                  sum_weight += a[i];
31                  sum_value += c[i];
32              }
33          if (sum_weight <= m)
34              ans = max(ans, sum_value);
35      }
36      In ra answer
37  }
```

Thuật toán 2

- Chia tập đồ vật làm hai phần A và B. Mỗi cách chọn lấy các đồ vật tương ứng với một cách lấy bên A kết hợp với một cách lấy bên B.
- Ý tưởng chính ở đây là lưu trữ hết các cách lấy bên B và sắp xếp trước theo một thứ tự. Sau đó với mỗi cách lấy bên A, ta có thể tìm kiếm nghiệm tối ưu bên B một cách nhanh chóng.
- Giả sử cách lấy tối ưu là lấy m_A bên A và m_B bên B, ta sẽ xét tuần tự từng m_A một và tìm kiếm nhị phân m_B.
- Độ phức tạp $O(2^A \times \log(2^B) + 2^B) = O(2^{n/2} \times n)$ nếu chọn |A| = |B| = n / 2

Code 2

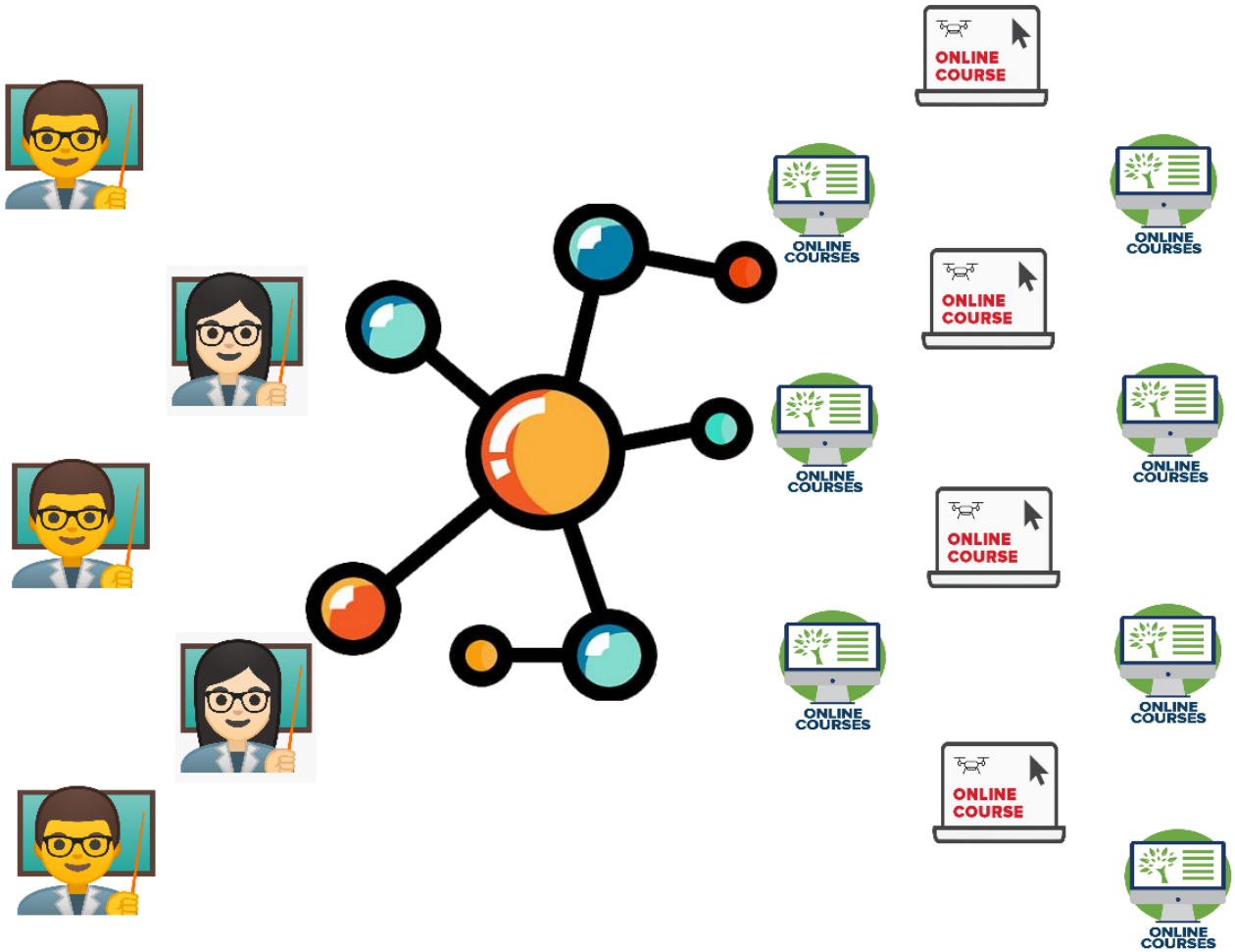
- S1, S2 là tập các cách lấy bên A, bên B có 2 tham số là khối lượng (w), giá trị (v) và đã được sort lại theo tham số khối lượng (w).
- $maxValue[j] = max\{S2[0].w, S2[1].w, ..., S2[j].w\}$

Code 2

```
39
40 int get_position(int S2[], int max_wend{itemize}ght) {
41     // tra ve vi tri cuoi cung cua S2 ma co S2.w <= max_weight
42     // tra ve -1 trong truong hop S2[0].w > max_weight
43 }
44
45 int main {
46     Nhap n, m
47     Nhap mang a, c
48     Tinh S1, S2 theo code 1
49     answer = -INF;
50     for(int i = 0; i < S1.size(); i++){
51         // lay vi tri lon nhat co the
52         j = get_position(S2, m - S1[i].w)
53         if (j != -1){
54             answer = max(answer, S1[i].v + max_value[j]);
55         }
56     }
57     In ra answer
58 }
```

BCA

- Mỗi đầu học kỳ các bộ môn phải thực hiện phân công giảng dạy đều cho các giảng viên.
- Có n khóa học và m giáo viên, mỗi giáo viên có danh sách các khóa có thể dạy.
- Có danh sách các khóa học không thể để cùng một giáo viên dạy do trùng giờ.
- Load của một giáo viên là số khóa học phải dạy của giáo viên đó.
- **Yêu cầu :** Tìm cách xếp lịch cho giáo viên sao cho Load tối đa của các giáo viên là tối thiểu.


ONLINE COURSE
ONLINE COURSES

Thuật toán

- Sử dụng thuật toán vét cạn, duyệt toàn bộ khóa học, xếp giáo viên dạy khóa học đó.
- Chọn khóa học chưa có người dạy cho giáo viên dạy ít nhất để phân công trước.
- Nếu phân công cho giáo viên A môn X, mọi môn học trùng lịch với môn X không thể được dạy bởi giáo viên A sau này.
- Nếu maxLoad hiện tại lớn hơn minLoad tối ưu thu được trước đó thì không duyệt nữa.

Code

```
59 bool cmp(int p, int q) {return load[p] < load[q];}
60 int get_ans() { return *max_element(load + 1, load + n + 1); }
61 void arrange(int i) {
62     if (get_ans() >= ans)  return;
63     if (i > n) {
64         ans = min(ans, get_ans()); // Cap nhat ket qua
65         return;
66     }
67     vector < int > index; // cac giao vien co the day course i
68     for (int j = 1; j <= m; j++) {
69         // Giao vien j day duoc course i
70         // Giao vien j khong day course conflict voi course i truoc do
71         if (teach[j][i] == 1 && cant_assign[j][i] == 0)
72             index.push_back(j);
73     }
74     // sort cac giao vien theo so luong course giang day
75     sort(index.begin(), index.end(), cmp);
76     for (int j : index) {
77         ** cap nhat lai cac mang load va cant_assign **
78         arrange(i + 1);
79         ** tra de quy **
80     }
81 }
```

Code

```
82  int main() {
83      cin >> m >> n;
84      for (int j = 1; j <= m; j++) {
85          int k;
86          cin >> k;
87          while (k--) {
88              int p;
89              cin >> p;
90              teach[j][p] = 1;
91          }
92      }
93      int q;
94      cin >> q;
95      while (q--) {
96          int p1, p2;
97          cin >> p1 >> p2;
98          if (p1 > p2) {
99              swap(p1, p2);
100         }
101         conflict[p1][p2] = 1;
102     }
103     ans = INF;
104     arrange(1);
105     cout << ans << endl;
106 }
```