Thuật toán ứng dụng

BÀI TẬP LẬP TRÌNH

Giảng viên: Ban Hà Bằng
TA: Phan Trung Kiên

SOICT — HUST

04/2020

Outline

03. CVRP OPT

- Một đội xe tải gồm K xe có tải trọng là Q cần được xếp lịch để giao những kiện hàng Pepsi từ một kho chứa 0 đến các khách hàng $1, 2, \ldots, n$. Mỗi khách hàng i yêu cầu $d[i]$ kiện hàng.
- Vấn đề: Với mỗi xe tải, một tuyến đường từ kho đến các khách hàng và trở về như sau:
  - Mỗi khách hàng chỉ được thăm đúng một lần bởi một xe tải
  - Số kiện hàng mà mỗi khách hàng yêu cầu không vượt quá tải trọng xe Q
- Mục tiêu
  - Tìm cách định tuyến sao cho tổng đường đi là ngắn nhất

Thuật toán

- $K = 1, Q = INFINITY$: Chính là bài toán TSP
- $K = 1$: Là bài TSP nhưng có thêm chặn Q, thậm chí dễ hơn
- $K > 1$: Cần duyệt các cách phân tập $1..N$ thành K tập con khác rỗng, mỗi tập con chính là 1 bài TSP độc lập nhau.
- Mỗi phần tử có thể thuộc 1 trong K tập, ta duyệt hết K^N cách phân tập

Thuật toán

- Thay vì với mỗi cách chia tập con, ta đi tìm đường đi ngắn nhất cho mỗi tập con, ta có thể tìm đường đi ngắn nhất trước
- u là đỉnh hiện tại
- m là tập đỉnh đã đi qua
- $F[m]$ đường đi tối ưu qua các đỉnh tập m và trở về 0
- q là số hàng còn lại trên xe

Cài đặt

```
1  const int N = 13;
2  int n, k, q, d[N], c[N][N];
3  int f[1 << N], s[5], res = 1e9;
4
5  void go(int u, int m, int q, int g) {
6      f[m] = std::min(f[m], g + c[u][0]);
7      for (int i = 1; i <= n; ++i)
8          if (q >= d[i] && !(m >> i - 1 & 1))
9              go(i, m | 1 << i - 1, q - d[i], g + c[u][i]);
10 }
11
12 void opt(int u) {
13     if (u == n) {
14         int t = 0;
15         for (int i = 0; i < k; ++i) t += f[s[i]];
16         res = std::min(res, t);
17     } else {
18         for (int i = 0; i < k; ++i)
19             if (f[s[i] | 1 << u] < res)
20                 s[i] ^= 1 << u, opt(u + 1), s[i] ^= 1 << u;
21     }
22 }
23
24 int main() {
25     scanf("%d %d %d", &n, &k, &q);
26     for (int i = 1; i <= n; ++i) scanf("%d", d + i);
27     for (int i = 0; i <= n; ++i)
28         for (int j = 0; j <= n; ++j) scanf("%d", &c[i][j]);
29     memset(f, 0x3F, sizeof f);
30     go(0, 0, q, 0);
31     opt(0);
32     printf("%d", res);
33 }
```

04. PIE

- Có N cái bánh và $F + 1$ người.
- Mỗi cái bánh có hình tròn, bán kính r và chiều cao là 1.
- Mỗi người chỉ được nhận một miếng bánh từ một chiếc bánh.
- Cần chia bánh sao cho mọi người có lượng bánh bằng nhau (có thể bỏ qua vụn bánh).
- Tìm lượng bánh lớn nhất mỗi người nhận được.

Thuật toán

- Gọi $p[i]$ là số người ăn chiếc bánh thứ i. Lượng bánh mỗi người nhận được là $min_i\{\frac{V[i]}{p[i]}\}$ với $V[i]$ là thể tích của chiếc bánh thứ i.
- **Cách 1 - Tìm kiếm theo mảng p:** Tìm kiếm vét cạn mọi giá trị của p.

Thuật toán

- Gọi $p[i]$ là số người ăn chiếc bánh thứ i. Lượng bánh mỗi người nhận được là $min_i\{\frac{V[i]}{p[i]}\}$ với $V[i]$ là thể tích của chiếc bánh thứ i.
- **Cách 1 - Tìm kiếm theo mảng p:** Tìm kiếm vét cạn mọi giá trị của p.
- **Cách 2 - Tìm kiếm theo lượng bánh mỗi người nhận được:** Thử từng kết quả, với mỗi kết quả, kiểm tra xem có thể chia bánh cho tối đa bao nhiêu người.
- **Tối ưu cách 2:** Sử dụng thuật toán tìm kiếm nhị phân để tìm kiếm kết quả.

Code

```
1     sort(r, r + N);
2
3         double lo = 0, hi = 4e8, mi;
4
5         for(int it = 0; it < 100; it++){
6             mi = (lo + hi) / 2;
7
8             int cont = 0;
9
10            for(int i = N - 1;
11            i >= 0 && cont <= F; --i)
12                cont += (int)
13                floor(M_PI * r[i] / mi);
14
15            if(cont > F) lo = mi;
16            else hi = mi;
17        }
```

04. EKO

- Cho n cái cây có chiều cao khác nhau $a_1, a_2, ...a_n$
- Có thể thực hiện một phát cắt độ cao h với tất cả các cây.
- Số lượng gỗ thu được là phần chóp của các cây cao hơn h.
- Tìm h lớn nhất có thể để số lượng gỗ thu được lớn hơn m.
- VD:
  - có 4 cây $20, 15, 10, 17$.
  - chọn $h = 15 \rightarrow$ số lượng gỗ thu được ở mỗi cây là $5, 0, 0, 2$. tổng là 7.
  - vậy ta thu được 7 mét gỗ.

Thuật toán

- **Thuật toán 1:** tìm tất cả các giá trị $h \in \{0, max(a[i])\}$. Với mỗi h, tính số lượng gỗ thu đượ. ĐPT: $O(max(a[i]) * n)$.
- **Thuật toán 2:** chặt nhị phân giá trị h.

Code

```
18      long long count_wood(int height) {
19          long long ret = 0;
20          for (int i = 1; i <= n; i++)
21              if ( a[i] > height )
22                  ret += a[i] - height;
23          return ret;
24      }
25
26      int l = 0, r = max(r,a[i]);
27
28      while (l < r-1) {
29          int mid = (l+r)/2;
30          if (count_wood(mid) >= m ) l = mid;
31          else r = mid;
32      }
33      cout << l;
```

04. AGGRCOW

- Có N chuồng bò và C con bò.
- Chuồng bò thứ i có tọa độ là x_i.
- Cần xếp các con bò vào các chuồng sao cho khoảng cách nhỏ nhất giữa 2 con bò bất kỳ là lớn nhất.

Thuật toán

- **Thuật toán 1:** Duyệt vét cạn từng con bò vào từng chuồng rồi tính khoảng cách ngắn nhất, $O(N^C \times C)$.

Thuật toán

- **Thuật toán 1:** Duyệt vét cạn từng con bò vào từng chuồng rồi tính khoảng cách ngắn nhất, $O(N^C \times C)$.
- **Thuật toán 2:** Duyệt giá trị kết quả bài toán d. Mỗi d, xếp các con bò vào chuồng 1 cách tham lam sao cho con bò sau cách con bò trước ít nhất d đơn vị. Nếu xếp đủ C con bò thì d là một giá trị hợp lệ. Tìm d lớn nhất. $O(max(x_i) \times N)$.

Thuật toán

- **Thuật toán 1:** Duyệt vét cạn từng con bò vào từng chuồng rồi tính khoảng cách ngắn nhất, $O(N^C \times C)$.
- **Thuật toán 2:** Duyệt giá trị kết quả bài toán d. Mỗi d, xếp các con bò vào chuồng 1 cách tham lam sao cho con bò sau cách con bò trước ít nhất d đơn vị. Nếu xếp đủ C con bò thì d là một giá trị hợp lệ. Tìm d lớn nhất. $O(max(x_i) \times N)$.
- **Thuật toán 3:** Tìm kiếm nhị phân với giá trị d. $O(\log max(x_i) \times N)$.

Code

```
34      sort(x + 1, x + n + 1);
35      int low = -1, high = (int)1e9 + 10;
36      while (high - low > 1) {
37          int mid = (low + high) / 2;
38          int num = 0;
39          int last = (int)-1e9;
40          for (int i = 1; i <= n; i++) {
41              if (x[i] >= x[i - 1] + last) {
42                  num++;
43                  last = x[i];
44              }
45          }
46          if (num >= C) low = mid;
47          else high = mid;
48      }
49      cout << low << endl;
```