ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Thực hành Thuật toán ứng dụng

Tuần 2

Integer Linear Equation

- Cho 2 số nguyên dương *n*, *M* và *n* số nguyên dương $a_1, a_2, ..., a_n$.

  Tính số nghiệm nguyên dương của phương trình: $a_1X_1 + a_2X_2 + . . . + a_nX_n = M$

- **Input**
  - Dòng 1: n và M
  - Dòng 2: $a_1, a_2, ..., a_n$
- **Output**
  - Số nghiệm nguyên dương

Integer Linear Equation

- Example

stdin	stdout
3 5 1 1 1	6

$1X_1 + 1X_2 + 1X_3 = 5$

-> nghiệm nguyên dương của X

(1,1,3)
(1,3,1)
(1,2,2)
(2,1,2)
(2,2,1)
(3,1,1)

Integer Linear Equation - Hint

Áp dụng tìm kiếm quay lui để duyệt các phương án thỏa mãn ràng buộc đặt ra

- Xét các biến từ trái qua phải $X_1, X_2, \ldots, X_{k-1}, X_k, X_{k+1}, \ldots, X_n$.
- Giả sử đã gán được giá trị cho $X_1, X_2, \ldots, X_{k-1}$. Ta xét X_k
- X_k nhận các giá trị từ 1 đến $M - a_1X_1 + a_2X_2 + \ldots + a_{k-1}X_{k-1} - (a_{k+1} + a_{k+2} + \ldots + a_n))/a_k$
- Biến phụ trợ
- f : tổng $a_1X_1 + a_2X_2 + \ldots + a_kX_k$ đối với các biến đã được gán giá trị
- f : được cập nhật tích lũy
- Mảng $t[1\ldots n]$ trong đó $t[k] = a_1 + a_2 + \ldots + a_k$
- Try(k): thử giá trị mới cho $x[k]$
- Với mỗi giá trị v (chạy từ 1 đến $(M - f - (t[n] - t[k]))/a_k$) gán cho $x[k]$, thực hiện
- Cập nhật: $f = f + a_kX_k$
- Nếu $k < n$ thì gọi tiếp Try(k+1)
- Ngược lại thì ghi nhận 1 lời giải

02. Phân công giảng dạy (BCA)

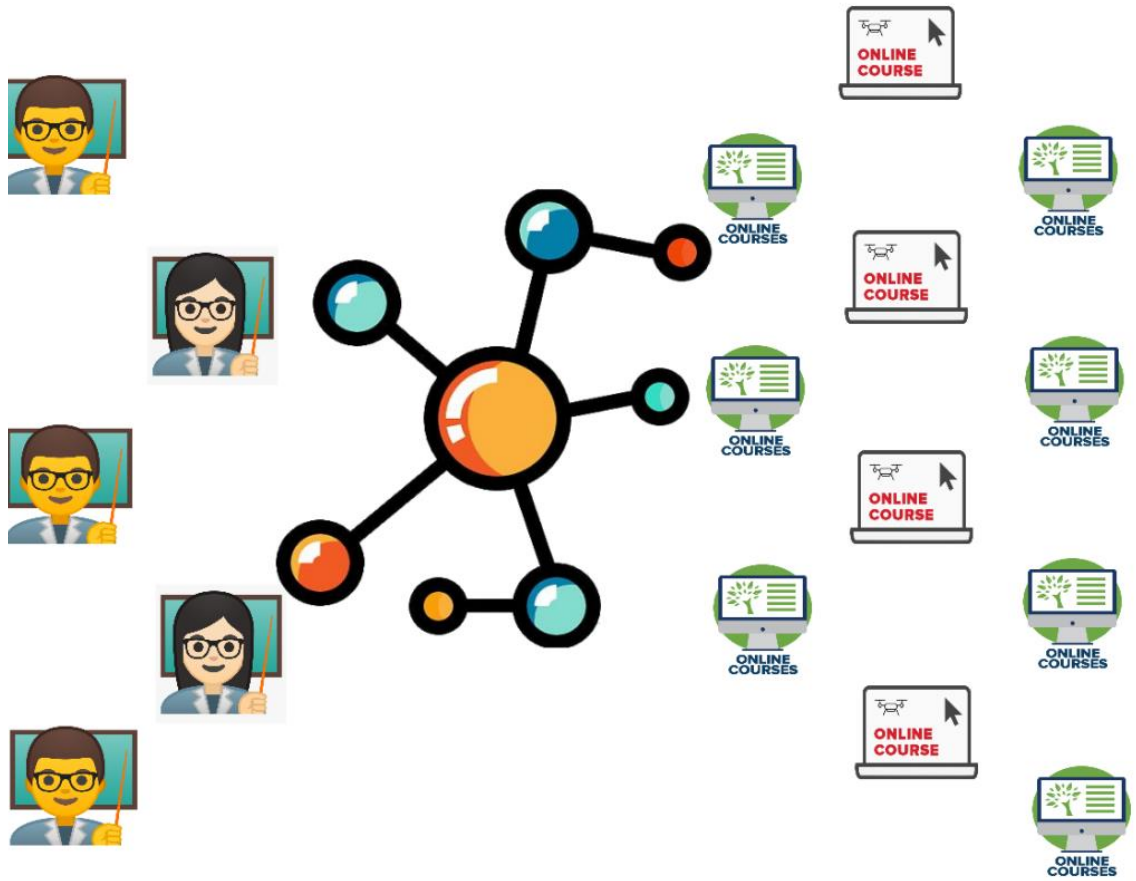
- 2.1 Lịch sử bài toán
- 2.2 Phát biểu bài toán
- 2.3 Ứng dụng
- 2.4 Thuật giải và cài đặt

3.1 Lịch sử bài toán

- Bài toán gán việc là bài toán tổng quát
  - Trong quá trình phân công công việc, thì mỗi công việc sẽ tạo ra các giá trị khác nhau, và mỗi người lao động có khi được phân công công việc sẽ được trả chi phí khác nhau theo trình độ của họ và công việc không vượt quá khả năng của người lao động.
  - Việc phân công hiệu quả khi chi phí là tối thiểu nhưng lợi nhuận lại tối đa đồng thời các công việc được giao phải phù hợp với người lao động.
- Khi số lượng người lao động là bằng 01 thì đó là bài toán lựa chọn các công việc để đối đa lợi nhuận.

3.2 Phát biểu bài toán 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 thể dạy nhiều khóa học khác nhau.
  - Mỗi một giáo viên không thể dạy hai khóa học có giờ trùng nhau.
  - 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.



3.3 Ứng dụng

- Áp dụng trong các bài toán lập lịch phân công công việc
- Áp dụng vào lập lịch trình cất cánh hạ cánh trên các đường băng

Balanced Course Assignment (BCA)

Input
- Dòng 1: chứa hai số nguyên m và n (1≤m≤10, 1≤n≤30)
  - *n khóa học* và *m giáo viên*
- Dòng i + 1: chứa số nguyên dương k và k số nguyên dương cho biết các khóa học mà giáo viên i có thể dạy (∀i = 1,…, m)
- Dòng m + 2: chứa số nguyên k
- Dòng i + m + 2: chứa hai số nguyên i và j cho biết hai khóa học trùng giờ nhau (∀i = 1,…, k)

Output
- Đầu ra chứa một số duy nhất là tải (số lớp) lớn nhất của giáo viên trong lời giải tìm được và giá trị -1 nếu không tìm thấy lời giải.

Dòng m + 2

stdin	stdout
4 12 5 1 3 5 10 12 5 9 3 4 8 12 6 1 2 3 4 9 7 7 1 2 3 5 6 10 11 5 1 2 1 3 1 5 2 4 2 5	3

3.4 Thuật giải và cài đặt

- Cách 1:
- Sử dụng *thuật toán vét cạn*, duyệt toàn bộ khóa học, xếp giáo viên dạy khóa học đó.
- 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.

Balanced Course Assignment (BCA): Hint

Cách 2

- Sử dụng backtracking để duyệt qua tất cả các phương án, kết hợp Branch and Bound để bỏ qua các tính toán dư thừa
- Biểu diễn lời giải: $x[1…n]$ trong đó $x[i]$ là giáo viên được phân công dạy môn i (i = 1, 2…, n)
- res: giá trị hàm mục tiêu tối ưu
- Các cấu trúc dữ liệu phụ trợ:
  - load[t]: số môn được phân công cho giáo viên t (t = 1, …, m).
  - load[t]: được tích lũy dần trong quá trình duyệt
- Branch and Bound:
  - Try(i): thử tất cả các giá trị (giáo viên) cho môn x[i]
    - Với mỗi giá trị (giáo viên) t được gán cho $x[i]$, thực hiện:
      - Kiểm tra giáo viên t có dạy môn nào trùng lịch với môn i không
      - Cập nhật: load[t] = load[t] + 1
      - Nếu load[t] < res thì gọi tiếp Try(i+1)
      - Ngược lại thì thuật toán quay lui để duyệt giá trị khác cho $x[i]$

03. Lập lộ trình xe bus (CBUS)

- 3.1 Lịch sử bài toán
- 3.2 Phát biểu bài toán
- 3.3 Ứng dụng
- 3.4 Thuật giải và cài đặt

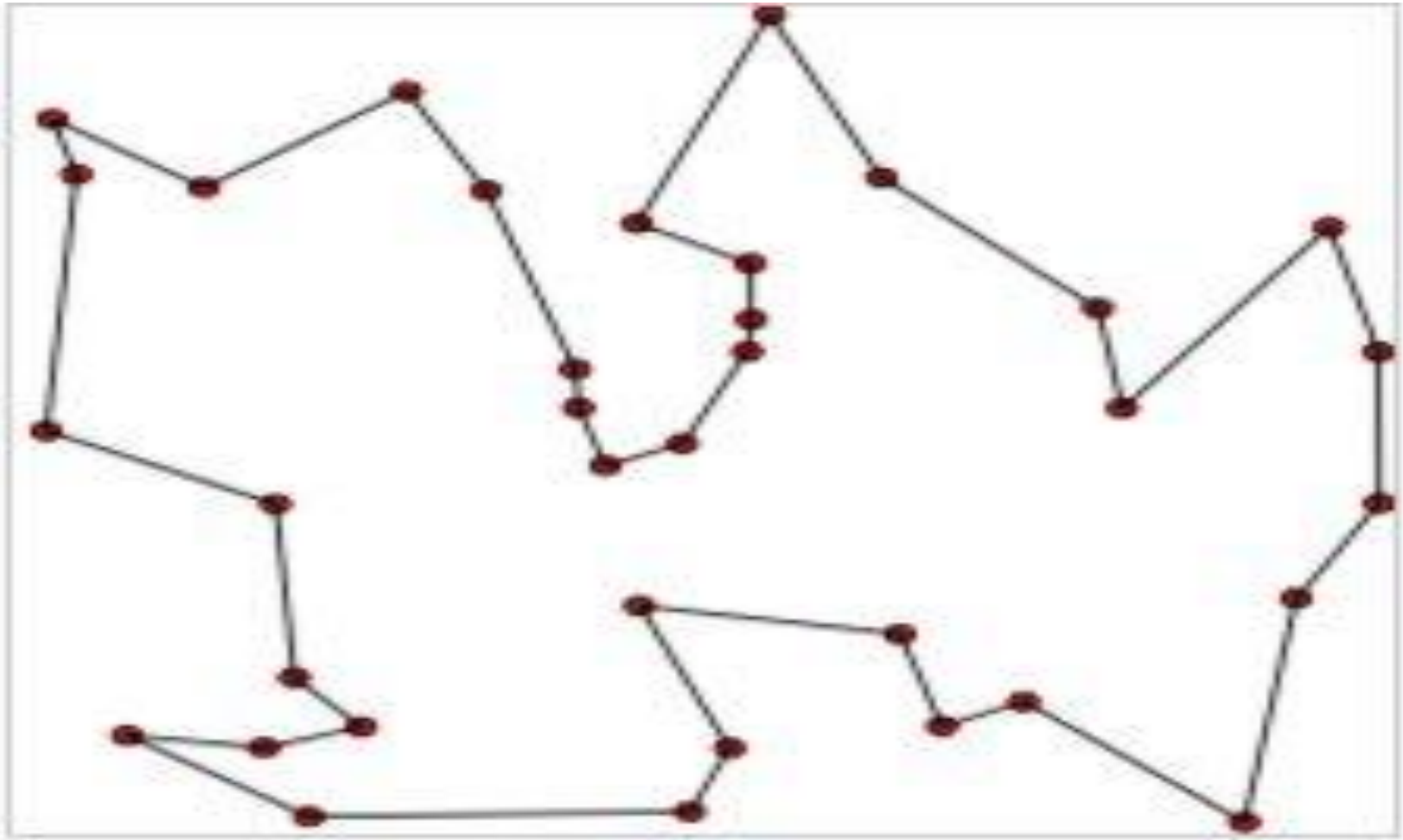
3.1 Lịch sử bài toán

- Bài toán định tuyến cho một tập các phương tiện đi để thỏa mãn phục vụ các khách hàng sao cho tối ưu về đường đi hay thời gian,... được nhắc đến trong nghiên cứu của George Dantzig và John Ramser vào năm 1959, trong đó đề xuất thuật toán đầu tiên được viết và được áp dụng cho việc giao xăng dầu.
- Bài toán này là khái quát của bài toán "Người du lịch"

3.2 Phát biểu bài toán

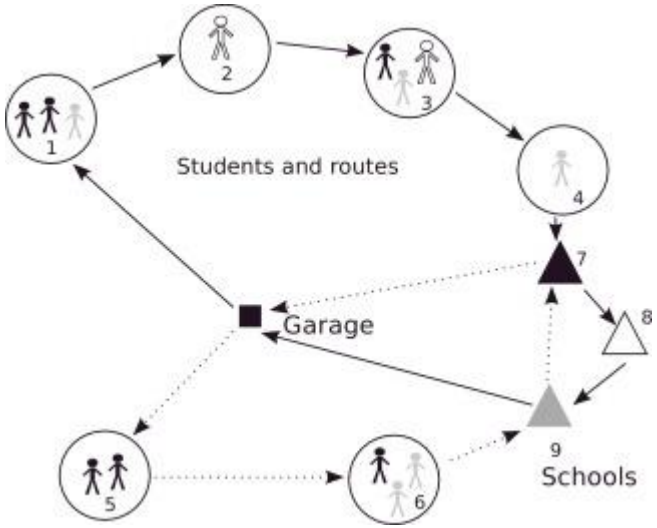
- Có n hành khách 1, 2,..., n, hành khách i cần di chuyển từ địa điểm i đến địa điểm j
- Xe khách *xuất phát ở địa điểm 0* và có thể *chứa tối đa k hành khách*
- Cho ma trận c với *c(i, j)* là khoảng cách di chuyển từ địa điểm i đến địa điểm j
- **Yêu cầu:** Tính khoảng cách ngắn nhất để xe khách phục vụ hết n hành khách và quay trở về địa điểm 0
- Lưu ý: Ngoại trừ địa điểm 0, các địa điểm khác chỉ được thăm tối đa 1 lần

3.2 Phát biểu bài toán



3.3 Ứng dụng

- Thiết kế tuyến xe bus cho trường học



- Thiết kế tuyến xe bus trong thành phố

CBUS

- Có n hành khách 1, 2,…, n. Hành khách thứ i muốn đi từ điểm *i* đến điểm *i* + *n* (i = 1,2,…, n).
- Có một xe buýt đặt tại điểm 0 và có *k* vị trí ghế ngồi (có nghĩa là bất cứ lúc nào cũng có nhiều nhất *k* hành khách trên xe buýt).
- Bạn được cung cấp ma trận khoảng cách *c* trong đó c (i, j) là khoảng cách di chuyển từ điểm i đến điểm j (i, j = 0,1,…, 2n).
- Tính tuyến đường ngắn nhất của xe buýt, phục vụ n hành khách và quay trở lại điểm 0.

Input

- Dòng 1 chứa n và k (1 ≤ n ≤11,1 ≤ k ≤10)
- Dòng i + 1 (i = 1, 2,…, 2n + 1) chứa dòng thứ (i - 1) của ma trận c (các hàng và cột được đánh chỉ số từ 0, 1, 2, .., 2n).

Output

- Dòng duy nhất chứa độ dài của tuyến đường ngắn nhất.

CBUS

- Example

stdin	stdout
3 2 0 8 5 1 10 5 9 9 0 5 6 6 2 8 2 2 0 3 8 7 2 5 3 4 0 3 2 7 9 6 8 7 0 9 10 3 8 10 6 5 0 2 3 4 4 5 2 2 0	25

Input

- Dòng 1 chứa n và k ($1 \le n \le 11, 1 \le k \le 10$)
- Dòng i + 1 (i = 1, 2,…, 2n + 1) chứa dòng thứ (i - 1) của ma trận c (các hàng và cột được đánh chỉ số từ 0, 1, 2, .., 2n).

Output

- Dòng duy nhất chứa độ dài của tuyến đường ngắn nhất.

CBUS - Hint

- Áp dụng kỹ thuật quay lui để duyệt tất cả các phương án, kết hợp Branch and Bound để loại bỏ các tính toán dư thừa
- Biểu diễn lời giải: $x_1, x_2, \ldots, x_{2n}$ là chuỗi $2n$ điểm (hoán vị của 1, 2, . . ., $2n$) đón – trả của hành khách trong hành trình xe bus.
- Biến phụ trợ
- Mảng đánh dấu *appear*[1..2*n*], trong đó *appear*[*v*] = true có nghĩa giá trị *v* (từ 1 đến 2*n*) đã xuất hiện trong hành trình bộ phận.
- load: ghi nhận số hành khách đang có mặt trên xe (được tích lùy dần trong quá trình duyệt)

- Hàm Try(k) thử giá trị cho *x*[*k*]
- Với mỗi giá trị v hợp lệ gán cho x[k], thực hiện
- Cập nhật: load = load + 1 nếu v <= n (điểm đón) và load = load – 1 nếu v > n (điểm trả)
- cập nhật mảng đánh dấu appear[v] = true
- Nếu k = 2n thì ghi nhận một phương án, so sánh với phương án tốt nhất đã có và thực hiện cập nhật kỷ lục
- Ngược lại, gọi tiếp Try(k+1)
- Áp dụng phương pháp đánh giá cận dưới (giống bài toán TSP) để tăng tốc độ duyệt, bỏ qua các nhánh tính toán thừa