**TRƯỜNG ĐẠI HỌC THỦY LỢI**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

----000----



**BÀI TIỂU LUẬN**

|  |  |
| --- | --- |
| Môn học: | Toán rời rạc |
| Tên bài toán: | Bài toán thuê thợ |
| Giáo viên hướng dẫn | Nguyễn Huy Đức |
| Lớp: | 62TH-NB |
| Tên sinh viên | Trương Anh Dương |
| Mã số sinh viên | 2051060475 |

Hà Nội, 2021

1. BÀI TOÁN THUÊ THỢ

* Có n người thợ được đánh số từ 1 đến n, có n công việc được đánh số từ 1 đến n. Biết rằng mỗi người thợ có thể làm tất cả các công việc. Biết bảng giá chi phí thuê thợ được cho bởi ma trận a[n,n] với kích thước n.n, trong đó a[i,j] là số tiền thuê người thợ i làm công việc j (i=1..n, j=1..n).
* Viết chương trình nhập vào số nguyên n(1<=n<=100) và các giá trị của ma trận a, dữ liệu được cho trong file văn bản THỦETHO.INP: dòng đầu ghi số n, n dòng sau mỗi dòng ghi n số là các giá trị của ma trận a.

1. Liệt kê và đếm số cách phân công(số phương án phân công) sao cho mỗi thợ làm đúng 1 việc và mỗi việc chỉ do 1 thợ làm.
2. Tìm một cách phân công mỗi thợ làm đúng 1 việc và mỗi việc chỉ do 1 thợ làm sao cho tổng chi phí thuê thợ là nhỏ nhất. (phương án phân công để có tổng chi phí nhỏ nhất gọi là phương án tối ưu, tổng chi phí nhỏ nhất tìm được gọi là giá trị tối ưu của bài toán).

Kết quả: Viết ra màn hình(hoặc file THUETHO2.OUT): Dòng đầu tiên ghi tổng chi phí nhỏ nhất tìm được. N dòng sau mỗi dòng ghi 2 số i, j với ý nghĩa: thợ i được phân công làm việc j. Ví dụ:

|  |  |
| --- | --- |
| THUETHO.INP | THUETHO2.OUT |
| 5  2 7 1 8 2  15 9 0 5 9  2 10 19 9 10  17 17 4 4 4  2 11 13 4 18 | 17  1 2 7  2 3 0  3 1 2  4 5 4  5 4 4 |

1. **LỜI GIẢI BÀI TOÁN:**

Phương án (cấu hình) của bài toán: là hoán vị của n phần tử {1, 2,…, n}, x[i] = j nghĩa là thợ i được phân công làm việc j.

1. **Đánh giá 1:**

* Tính sẵn giá trị nhỏ nhất của từng dòng trong bảng chi phí, lưu vào mảng mindong: mindong[i] là giá trị nhỏ nhất của dòng i.
* Dựa vào mảng trên tính sẵn mảng mintudong, với ý nghĩa mintudong[i] là giá trị nhỏ nhất của bảng từ dòng i đến dòng n(tính theo công thức truy hồi-Quy hoạch dộng)
* for(int i = 1; i<n; i++)

{

mindong[i]=a[i][1]; for (int j=2; j<=n; j++)

if (a[i][j]<mindong[i]) mindong[i]=a[i][j];

}

mintudong[n]=mindong[n]; for (int i=n-1; i>=1; i--)

mintudong[i]=min(mintudong[i+1],mindong[i]);

Khi xây dựng xong các thành phần x[1]…x[i] tức là đã bố trí công việc cho các thợ 1,2,..,i ta tính dần được chi phí đã thuê là S (gọi là giá của phương án bộ phận). Việc đánh giá để quyết định xây dựng tiếp thành phần x[i+1] hay không như sau

Khi còn n – i thành phần (người thợ) chưa được xây dựng (chưa được phân công), chúng ta đánh giá theo kiểu “làm non” như sau: nếu mỗi thành phần trong n – i thành phần còn lại từ x[i+1] đến x[n] nhận một giá trị trong bảng tại một dòng tương ứng, thì các giá trị ấy đều phải lớn hơn hoặc bằng giá trị nhỏ nhất trong các dòng từ i+1 đến n đã được tính sẵn đó là mintudong[i+1], có n – i dòng, vậy tổng các số với bất cứ cách chọn n – i số nào đều phải lớn hơn hoặc bằng (n – i)\* mintudong[i+1] tức là (n-i)\*mintudong[i+1] là một cận dưới của các tổng chi phí cho các thành phần x[i+1] đến x[n], hay s+(n-i)\*mintudong[i+1] là một cận dưới của giá trị các phương án giống nhau i thành phần đầu, chỉ khác nhau n – i thành phần sau, đây là một cận dưới cần tìm.

* Đánh giá nhánh cận: if (**s+(n-i)\*mintudong[i+1]<=Min**) Try(i+1);

Nhận xét: Cận dưới này vẫn còn quá nhỏ (chưa sát), vì vậy khả năng được cắt nhánh chưa nhiều(chỉ cắt nhánh khi bất đẳng thức đánh giá là sai, mà cận nhỏ quá thì ít khi bất đẳng thức sai), tuy nhiên tốc độ thực hiện chương trình đã được cải thiện đáng kể so với trường hợp không đánh giá nhánh cận.

Sau đây chúng ta cải tiến việc tìm cận dưới để có đánh giá hiệu quả hơn.

1. **Đánh giá 2:**

Thay vì tính các giá trị **mintudong[i]** như trên, ta đi **tính tổng các giá trị nhỏ nhất của các dòng từ i đến n**, lưu trữ sẵn trong mảng tongmintu, như sau:

tongmintu[n]= mindong[n]; for (int i=n-1; i>=1; i--)

tongmintu[i]=tongmintu[i+1]+mindong[i];

Ta có **s + tongmintu[i+1]** chắc chắn là một cận dưới và cận dưới này nhỏ hơn cận dưới của cách Đánh giá 1 bên trên.

**Đánh giá nhánh cận:** if (**s+tongmintu[i+1]<=Min**) Try(i+1);

Việc đánh giá này khá hiệu quả, cắt được nhiều nhánh, chương trình chạy nhanh hơn với các bộ dữ liệu lớn.

Bài toán nhóm em làm theo đánh giá 2.