CHƯƠNG 3.

BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT.

Những bài toán tìm đường đi trong các đồ thị (đặc biệt là tìm đường đi ngắn nhất) được kể là một trong những bài toán kinh điển, cổ trong lý thuyết đồ thị và có nhiều ứng dụng nhất.

3.1. ĐỊNH NGHĨA.

Cho G = (X, U) là một đồ thị có định giá; tương ứng với mỗi cung u=(i, j), có một chiều dài (hay trọng lượng) l(u) hay l_{ij}.
Bài toán tìm đường đi ngắn nhất giữa i và j là tìm một đường μ(i, j) từ i đến j sao cho :

$$l(\mu) = \sum_{u} l(u)$$

là ngắn nhất.

Diễn giải l(μ) : Chi chí vận chuyển, Chi phí xây dựng, thời gian cần thiết để đi khắp,...

CHÚ Ý. Bài toán tìm đường đi ngắn nhất tương tự với bài toán tìm đường đi dài nhất.

Những thuật toán khác nhau theo những tính chất sau đây :

- ♦ $l(u) \geq 0, \forall u \in U$.
- ♦ l(u) bằng nhau $\Leftrightarrow$ $l(u) = 1, \forall u \in U$.(Bài toán đường đi ngắn nhất theo số cung)
- ♦ G không có chu trình.
- ♦ G và l(u) bất kỳ.

Và loại bài toán sau được xét :

- Tìm đường đi ngắn nhất từ một đỉnh đến các đỉnh còn lại,
- Tìm đường đi ngắn nhất giữa các cặp đỉnh.

3.2. NGUYÊN LÝ TỐI ƯU.

Nguyên lý tối ưu phát biểu theo sự kiện là tập đường đi con của tập đường đi ngắn nhất là những đường ngắn nhất.

BỔ ĐỀ.

Xét đồ thị G = (X,U) và một hàm trọng lượng l : X x X $\rightarrow$ R, Cho C = « x_1, x_2,...,x_k » là đường đi ngắn nhất từ x_1 đến x_k và với mọi (i, j) sao cho $1 \le i \le j \le k$, Cho C_{ij} = « x_i, x_{i+1},...,x_j » là đường con của C từ x_i đến x_j. Khi ấy C_{ij} là một đường ngắn nhất từ x_i đến x_j.

Nguyên lý của những thuật toán tìm đường đi ngắn nhất :

- Một khoảng cách d(i) tương ứng với đỉnh x_i.
- Ở cuối thuật toán, khoảng cách này biểu diễn chiều dài ngắn nhất từ gốc đến đỉnh đang xét.

3.3. CÁC DẠNG CỦA BÀI TOÁN: TỪ MỘT ĐỈNH ĐẾN CÁC ĐỈNH CÒN LẠI.

Bài toán này còn được gọi là bài toán tìm đường đi ngắn nhất từ gốc duy nhất. Nhiều bài toán khác cũng có thể dùng thuật toán này để giải :

- Đường đi ngắn nhất đến đích duy nhất.
- Đường đi ngắn nhất từ cặp đỉnh cho trước.
- Đường đi ngắn nhất cho mọi cặp đỉnh (thuật toán gốc duy nhất từ mỗi đỉnh).

3.3.1. THUẬT TOÁN DIJKSTRA-MOORE (1959).

Giả thiết là các cạnh (cung) (l(u) ≥ 0). Giả sử G có n đỉnh đánh số thứ tự từ 1 tới n. Bài toán đặt ra là tìm đường đi ngắn nhất từ đỉnh 1 đến các đỉnh còn lại trong đồ thị.

Ký hiệu :

- n_0 = số phần tử chưa chọn;
- A = Ma trận kề biểu diễn đồ thị, có trọng lượng, được định nghĩa như sau :

$$A = [a_{i,j}] = \begin{cases} l(i,j) = \text{chiều dài của cạnh cung ứng } u=(i,j) \in U \\ \infty \quad u=(i,j) \notin U \\ 0, \quad i=j \end{cases}$$

- Pr(p) = đỉnh trước đỉnh p theo đường đi ngắn nhất từ gốc đến đỉnh p.
- d = khoảng cách ngắn nhất từ gốc đến các đỉnh còn lại trong đồ thị. Qui ước ∞ cho các đỉnh không có đường đi từ gốc đến nó.
- Mark = Tập đỉnh đã đánh dấu (đã xét rồi), định nghĩa như sau :

$$Mark[i] = \begin{cases} 1, \text{ nếu đỉnh đã xét rồi,} \\ 0, \text{ ngược lại.} \end{cases}$$

NGUYÊN LÝ THUẬT TOÁN.

1. **Khởi tạo** : Xuất phát từ đỉnh 1 ; $n_0 = n - 1$:
   Pr = [1,1,...1]
   d = a[1,j], j=1..n (Dòng đầu của ma trận kề A)
   Mark = [1,0...0]
2. **Ở mỗi bước lặp**, chọn đỉnh đánh dấu là đỉnh có độ dài ngắn nhất trong những đỉnh chưa đánh dấu, nghĩa là chọn đỉnh **k** sao cho :
   - d[k] = Min {d[i] : Mark[i]= 0 } ;
   - Mark[k]=1.
   - Cập nhật lại d[j], Pr[j] với những đỉnh j chưa đánh dấu (Mark[j]=0) theo công thức:
     - d[j] = d[k] + a[k,j] nếu d[j] > d[k] +a[k,j].
     - Pr[j] = k.

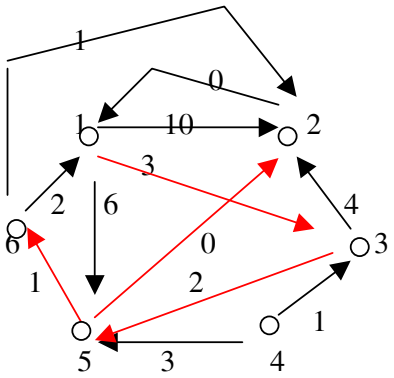
Nếu tất cả mọi đỉnh đã được chọn, nghĩa là $n_0 = 0$. Dừng. Nếu không , quay lại 2.

THỦ TỤC DIJKSTRA – MOORE ;

```
//Giả sử đã nhập ma trận chiều dài l theo dạng ma trận kề A
//Gán ban đầu cho d, Pr, Mark, n0 .
For (int j= 1; j≤ n ; j++)   { d[j] = a(1,j) ; pr[j]=1 ; Mark[j] = 0;}
Mark[1] =1 ; n0 = n-1 ;
WHILE (n0 > 0)
 {d[k] = Min {d[j] : Mark[j]= 0 } ;
 // Cập nhật lại n0 , d và Pr, Mark
   Mark[k] =1 ; n0 = n0 - 1 ;
   For (int j= 1; j≤ n ; j++) if (Mark [j] = 0) && (d[k]+ a[k,j] < d[j])
     { d[j] = d[k] +a[k,j] ; pr[j]=k}
 }
```

Độ phức tạp : $O(n^2)$ hay O(mlogn)

THÍ DỤ.



Ma trận kề A :

$$A = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 10 & 3 & \infty & 6 & \infty \\ 2 & 0 & \infty & \infty & \infty & \infty & \infty \\ 3 & \infty & 4 & 0 & \infty & 2 & \infty \\ 4 & \infty & \infty & 1 & 0 & 3 & \infty \\ 5 & \infty & 0 & \infty & \infty & 0 & 1 \\ 6 & 2 & 1 & \infty & \infty & \infty & 0 \end{array}$$

FIG.3.1. Đồ thị có định hướng, có trọng lượng.

- **Gán Ban đầu.** Cho Mark, d, Pr :
  Mark = [1, 0, 0, 0, 0, 0]
  d = [0, 10, 3, ∝, 6, ∝]
  Pr = [1, 1, 1, 1, 1, 1]
- **Bước 1.** Chọn đỉnh s_3. Cập nhật Mark, d, Pr :
  Mark = [1, 0, 1, 0, 0, 0]
  d = [0, 7, **3**, ∝, 5, ∝]
  Pr = [1, 3, **1**, 1, 3, 1]
- **Bước 2 .** Đỉnh hiện thời là s_3. Chọn đỉnh s_5. Cập nhật Mark, d, Pr :
  Mark = [1, 0, 1, 0, 1, 0]
  d = [0, 5, **3**, ∝, 5, 6]
  Pr = [1, 5, **1**, 1, 3, 5]
- **Bước 3 .** Đỉnh hiện thời là s_5 . Chọn đỉnh s_2. Cập nhật Mark, d, Pr :
  Mark = [1, 1, 1, 0, 1, 0]
  d = [0, 5, **3**, ∝, 5, 6]
  Pr = [1, 5, **1**, 1, 3, 5]
- **Bước 4 .** Đỉnh hiện thời là s_2 . Chọn đỉnh s_6. Cập nhật Mark, d, Pr :
  Mark = [1, 1, 1, 0, 1, 1]
  d = [0, 5, **3**, ∝, 5, 6]
  Pr = [1, 5, **1**, 1, 3, 5]

Thuật toán kết thúc vì đỉnh s_4, ta có $d[s_4]$ = Min {d[j] : Mark[j]= 0}= $d[s_4]$ = ∝.
Từ thuật toán , ta có kết quả sau :

d = [0, 5, **3**, ∝, 5, 6]
Pr = [1, 5, **1**, 1, 3, 5]

- Đường đi ngắn nhất từ s_1 đến s_2 : $s_1 \to s_3 \to s_5 \to s_2$ và độ dài là **5**
- Đường đi ngắn nhất từ s_1 đến s_3 : $s_1 \to s_3$ và độ dài là **3**
- Đường đi ngắn nhất từ s_1 đến s_5 : $s_1 \to s_3 \to s_5$ và độ dài là **5**
- Đường đi ngắn nhất từ s_1 đến s_6 : $s_1 \to s_5 \to s_6$ và độ dài là **6**
- Không có đường đi ngắn nhất từ đỉnh s_1 đến s_4 ($d[s_4]$ = ∝) , vì không có đường nối từ s_1 đến s_4.

GHI CHÚ.

Giả thiết « Hàm trọng lượng không âm » là bắt buộc. Chẳng hạn, sử dụng thuật toán Dijktra-Moore cho đồ thị ở hình FIG.3.2, dẫn đến kết quả sai nếu ta chọn gốc là đỉnh s_1. Thật vậy, đầu tiên, ta chọn đỉnh s_2, ($s_1 \rightarrow s_2$) trong khi đó, đường đi ngắn nhất là đường đi từ đỉnh s_1 đến s_2 qua s_3.

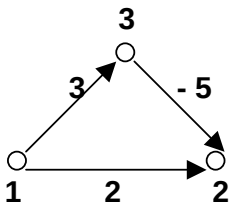


FIG. 3.2. Đồ thị có định hướng, có trọng lượng bất kỳ.

3.3.2. THUẬT TOÁN BELLMAN-FORD (1958-1962)

Sự hiện diện của dấu bất kỳ của trọng lượng (hay chiều dài) cho phép, chẳng hạn, có thể cải tiến chi phí hay lợi nhuận. Thuật toán DIJKSTRA-MOORE không cho phép xét tới những cạnh (cung) có trọng lượng không âm, vì trong trường hợp một cạnh được đánh dấu, thì ta không thể thay đổi gì cho những bước lặp tiếp theo. Thuật toán DIJKSTRA-MOORE còn được gọi là **gán nhãn cố định .**

Để giải quyết cho trường hợp đồ thị có trọng lượng bất kỳ, ta một xét thuật toán cho phép một đánh dấu chỉ được xác định hoàn toàn khi thuật toán kết thúc. Một kiểu thuật toán như vậy được gọi là **điều chỉnh nhãn.**

Thuật toán **BELLMAN-FORD** chỉ có giá trị cho các đồ thị không có chu trình, có trọng lượng bất kỳ.

Ký hiệu :

- Tập đỉnh được đánh số thứ tự từ 1 ..n.
- Pr(p) = đỉnh trước đỉnh p theo đường đi ngắn nhất từ gốc đến đỉnh p.
- d = khoảng cách ngắn nhất từ gốc đến các đỉnh còn lại trong đồ thị.
- Mark = Tập đỉnh đã đánh dấu (đã xét rồi), định nghĩa như sau :

$$\text{Mark}[i] = \begin{cases} 1, \text{ nếu đỉnh đã xét rồi,} \\ 0, \text{ ngược lại.} \end{cases}$$

Khoảng cách ngắn nhất từ gốc đến một đỉnh v chỉ được tính khi tất cả các phần tử trước của v ($\Gamma^-(v)$) đã được đánh dấu rồi. Một đỉnh bất kỳ, khi chưa đánh dấu, thì khoảng cách từ gốc đến đỉnh đó chưa biết (chưa tính).

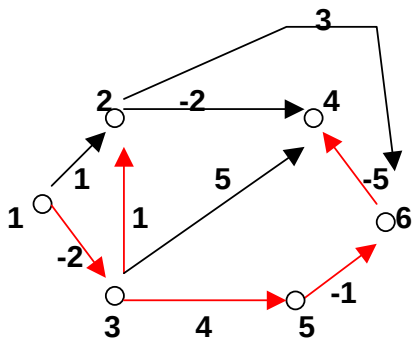
NGUYÊN LÝ THUẬT TOÁN

1. **Gán các giá trị ban đầu.**
   - Chọn đỉnh s_1 làm gốc.
   - Mark = [1,0...0] ; d[1] = 0 ; Pr[1] = 1.

2. **Ở mỗi bước lặp :**
   - Chọn đỉnh **k** chưa đánh dấu sao cho tất cả đỉnh trước của k đã đánh dấu rồi , nghĩa là : **Mark[k] = 0 và $\forall$ j $\in$ Γ^-(k) : Mark[j]= 0**
   - Cập nhật Mark : Mark[k] =1 ;
   - Tính d[k] = min { d[i] + a[i, k]: i $\in \Gamma^-$(k)}, và Pr[k] là chỉ số đạt min.

ĐỘ PHỨC TẠP : O(nm). $O(n^3)$ Cho các đồ thị đầy, i.e., những đồ thị mà $m \approx n^2$.

THÍ DỤ.



Gán ban đầu : Mark, d, Pr :
Mark = [1, 0, 0, 0, 0, 0},
d[1] = 0 ;
Pr [1] = 1
$\Gamma^-(2)=\{1,3\};\Gamma^-(3)=\{1\};\Gamma^-(4)=\{2,3,6\}$
$\Gamma^-(5)=\{3\}$; $\Gamma^-(6)=\{2,5\}$

FIG.3.1. Đồ thị có định hướng, có trọng lượng bất kỳ, không có chu trình, gốc đỉnh 1.

- **Bước 1.** Chọn đỉnh **3** vì $\Gamma^-(3)=\{1\}$. Cập nhật Mark[3], Tính d[3] và Pr[3] :
  Mark[3] = 1 ; d[3] = -2 ; Pr[3] = 1;

- **Bước 2.** Ở bước lặp này, ta có thể chọn đỉnh 5 (hay đỉnh 2).
  Cập nhật Mark[5], Tính d[5] và Pr[5] :
  Mark[5] = 1 ; d[5] = 2 ; Pr[5] = 3;

- **Bước 3.** Chọn đỉnh **2** . Cập nhật Mark[2], Tính d[2] và Pr[2] :
  Mark[2] = 1 ; d[2] = -1 ; Pr[2] = 3;

- **Bước 4.** Chọn đỉnh **6** . Cập nhật Mark[6], Tính d[6] và Pr[6] :
  Mark[6] = 1 ; d[6] = 1; Pr[6] = 5

- **Bước 5.** Chọn đỉnh **4** . Cập nhật Mark[4], Tính d[4] và Pr[4] :
  Mark[4] = 1 ; d[4] = - 4 ; Pr[4] = 6

Thuật toán kết thúc vì tất cả các đỉnh đã được chọn rồi.

Từ thuật toán , ta có kết quả sau :
d = [0, -1, -2, -4, 2, 1]
Pr = [1, 3, 1, 6, 3, 5]

- Đường đi ngắn nhất từ s_1 đến s_2 : $s_1 \to s_3 \to s_2$ và độ dài là **-1**
- Đường đi ngắn nhất từ s_1 đến s_3 : $s_1 \to s_3$ và độ dài là **-2**
- Đường đi ngắn nhất từ s_1 đến s_4 : $s_1 \to s_3 \to s_5 \to s_6 \to s_4$ và độ dài là - 4
- Đường đi ngắn nhất từ s_1 đến s_5 : $s_1 \to s_3 \to s_5$ và độ dài là **2**
- Đường đi ngắn nhất từ s_1 đến s_6 : $s_1 \to s_3 \to s_5 \to s_6$ và độ dài là 1

3.4. GIỮA TẤT CẢ CÁC CẶP ĐỈNH: THUẬT TOÁN FLOYD (1962).

Ta sẽ tính một ma trận khoảng cách n x n. Nếu tất cả chiều dài không âm ($l(u) \geq 0$) ta có thể áp dụng n lần thuật toán Dijktra-Moore cho mỗi đỉnh i. . Nếu đồ thị có chứa chiều dài âm ($l(u) < 0$) ta có thể áp dụng n lần thuật toán Bellman-Ford cho mỗiđỉnh i. Thuật toán Floyd có cách tiếp cận khác có lợi cho trường hợp ma trận dầy.

Ký hiệu :

- A : ma trận trọng lượng, được gán giá trị ban đầu như sau :

$$A[i,j] = \begin{cases} 0 & \text{nếu } i = j \\ l(i,j) & \text{nếu } (i,j) \in U \\ \infty & \text{ngược lại.} \end{cases}$$

- P : ma trận các đỉnh trước, được gán giá trị ban đầu như sau :
  P[i,j] = i, trong đó P[i,j] là đỉnh trước của đỉnh j trên đường đi từ gốc i đến j

Khi kết thúc thuật toán, ta có :

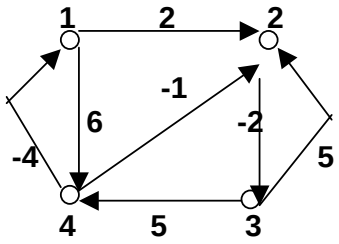
P[i,j] = đỉnh trước của j trên đường đi ngắn nhất từ gốc i đến đỉnh j, với chiều dài tương ứng là A[i,j].

THỦ TỤC FLOYD(L, P)

```
For (k =1;  k≤ n ; k++)
      For (i =1 ;i≤ n ; i++)
          For (j =1 ;j≤ n ; j++)
              If  (a[i,k] +  a[k,j]  <  a[i,j])
              { a[i,j] := a[i,k]  +  a[k,j]  ;  p[i,j] :=p[k,j] ;}
```


Độ phức tạp : $O(n^3)$.

THÍ DỤ.



Gán ban đầu : cho các ma trận A, P.

$$A_0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 2 & \infty & 6 \\ 2 & \infty & 0 & -2 & \infty \\ 3 & \infty & 5 & 0 & 5 \\ 4 & -4 & -1 & \infty & 0 \end{array} \qquad P_0 = \begin{array}{|cccc|} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{array}$$

Các bước lặp :

- **k =1.**

$$A_1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 2 & \infty & 6 \\ 2 & \infty & 0 & -2 & \infty \\ 3 & \infty & 5 & 0 & 5 \\ 4 & -4 & -2 & \infty & 0 \end{array} \qquad P_1 = \begin{array}{|cccc|} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 1 & 4 & 4 \end{array}$$

- **k = 2**

$$A_2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 2 & 0 & 6 \\ 2 & \infty & 0 & -2 & \infty \\ 3 & \infty & 5 & 0 & 5 \\ 4 & -4 & -2 & -4 & 0 \end{array} \qquad P_2 = \begin{array}{|cccc|} 1 & 2 & 3 & 4 \\ 1 & 1 & 2 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 1 & 2 & 4 \end{array}$$

- **k =3**

$$A_3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 2 & 0 & 5 \\ 2 & \infty & 0 & -2 & 3 \\ 3 & \infty & 5 & 0 & 5 \\ 4 & -4 & -2 & -4 & 0 \end{array} \qquad P_3 = \begin{array}{|cccc|} 1 & 2 & 3 & 4 \\ 1 & 1 & 2 & 3 \\ 0 & 2 & 2 & 3 \\ 0 & 3 & 3 & 3 \\ 4 & 1 & 2 & 4 \end{array}$$

- **k = 4**

$$A_4 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 2 & 0 & 5 \\ 2 & -1 & 0 & -2 & 3 \\ 3 & 1 & 3 & 0 & 5 \\ 4 & -4 & -2 & -4 & 0 \end{array} \qquad P_4 = \begin{array}{|cccc|} 1 & 2 & 3 & 4 \\ 1 & 1 & 2 & 3 \\ 0 & 2 & 2 & 3 \\ 4 & 1 & 3 & 3 \\ 4 & 1 & 2 & 4 \end{array}$$

Cách nhận biết đường đi ngắn nhất.

Để nhận được đường đi ngắn nhất từ s_1 đến s_j, ta sử dụng dòng thứ i của ma trận P. Chẳng hạn, ta muốn nhận được đường đi ngắn nhất $\mu : s_4 \rightarrow s_3$, ta tham khảo ma trận P như sau : P[4,3]=2 :s_2 là đỉnh trước của s_3 ; P[4,2]=1 : s_1 là đỉnh trước của s_2 ; P[4,1]=4 :s_4 là đỉnh trước của s_1.
Cuối cùng, kết quả là $\mu = s_4 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$.

Một trong ứng dụng của Thuật toán FLOYD là tìm đường đi giữa hai đỉnh. Thuật toán này được WARSHALL phát triển cùng năm (1962), và thuật toán thường mang tên FLOYD-WARSHALL ».

Ký hiệu :

❖ A = ma trận kề của đồ thị, được gán giá trị ban đầu như sau :

$$A[i,j] = \begin{cases} 1 & \text{nếu } (i, j) \in U \\ 0 & \text{ngược lại.} \end{cases}$$

❖ P = ma trận các đỉnh trước, được gán giá trị ban đầu như sau :

$$P[i,j] = \begin{cases} 0 & \text{nếu } a[i,j] = 0, \\ 1 & \text{ngược lại.} \end{cases}$$

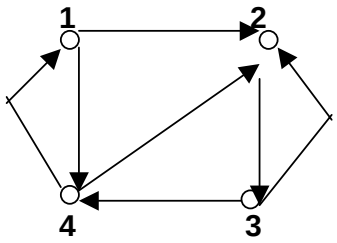
Khi kết thúc thuật toán :
P[i,j] = đỉnh trước của j trên đường đi từ đỉnh i đến đỉnh j (nghĩa là a[i,j]=1).

THỦ TỤC FLOYD-WARSHAL(A, P)

```
For (k =1 ;k≤ n ; k++)
    For (i =1 ;i≤ n ; i++)
        For (j =1 ;j≤ n ; j++)
            If  (a[i,j]  = =  0)
            {       a[i,j] = a[i,k] *a[k,j]  ; p[i,j] =p[k,j] }
```


Độ phức tạp : $O(n^3)$.

THÍ DỤ.



Gán ban đầu : cho các ma trận A, P.

A_0 =

	1	2	3	4
1	0	1	0	1
2	0	0	1	0
3	0	1	0	1
4	1	1	0	0

P_0 =

0	1	0	1
0	0	2	0
0	3	0	3
4	4	0	0

Các bước lặp :

- **k =1.**

A_1 =

	1	2	3	4
1	0	1	0	1
2	0	0	1	0
3	0	1	0	1
4	1	1	0	1

P_1 =

0	1	0	1
0	0	2	0
0	3	0	3
4	4	0	1

- **k = 2**

A_2 =

	1	2	3	4
1	0	1	1	1
2	0	0	1	0
3	0	1	1	1
4	1	1	1	1

P_2 =

0	1	2	1
0	0	2	0
0	3	2	3
4	4	2	1

- **k =3**

A_3 =

	1	2	3	4
1	0	1	1	1
2	0	1	1	1
3	0	1	1	1
4	1	1	1	1

P_3 =

0	1	2	1
0	3	2	3
0	3	2	3
4	4	2	1

- **k = 4**

A_4 =

	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1

P_4 =

4	1	2	1
4	3	2	3
4	3	2	3
4	4	2	1