ĐỀ THI GIỮA KỲ: 4h10 – 5h10

MÔN ĐỒ ÁN 1 – LẬP TRÌNH PYTHON

Bài 1 (2 điểm):

Viết chương trình tính sin(x) theo công thức xấp xỉ:

$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + ... + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$ với độ chính xác 0.00001. Tức là tính cho tới n sao cho: $\left|\frac{x^n}{n!}\right| < 0.00001$.

Bài 2 (2 điểm):

Viết hàm tính giá trị đa thức bậc n: $F(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$, hàm có 3 tham số là mảng số thực các hệ số a_i, giá trị n, x. (*Không được sử dụng hàm lũy thừa sẵn có*)

Bài 3 (2 điểm):

Viết hàm nhập, xuất ma trận vuông các số nguyên

–Hàm duyệt các phần tử trên đường chéo chính

–Hàm duyệt các phần tử thuộc tam giác trên đường chéo chính

–Hàm duyệt các phần tử thuộc tam giác dưới đường chéo chính

–Hàm duyệt các phần tử trên đường chéo phụ

–Hàm duyệt các phần tử thuộc tam giác trên đường chéo phụ

–Hàm duyệt các phần tử thuộc tam giác dưới đường chéo phụ

Bài 4 (1 điểm):

Đồ thị mà mỗi cạnh của nó được gán cho tương ứng với một số (nguyên hoặc thực) được gọi là đồ thị có trọng số. Số gán cho mỗi cạnh của đồ thị được gọi là trọng số của cạnh. Tương tự như đồ thị không trọng số, có nhiều cách biểu diễn đồ thị có trọng số trong máy tính. Đối với đơn đồ thị thì cách dễ dùng nhất là sử dụng ma trận trọng số:

Giả sử đồ thị G = (V, E) có n đỉnh. Ta sẽ dựng ma trận vuông C kích thước n x n. Ở đây:

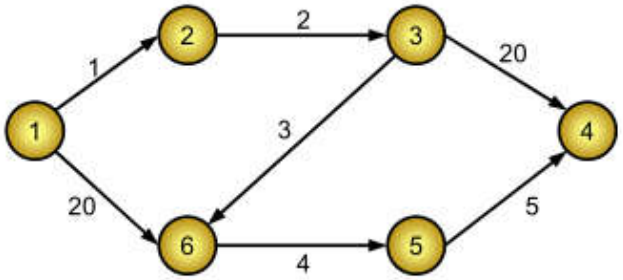
- Nếu (u, v) ∈ E thì C[u, v] = trọng số của cạnh (u, v)
- Nếu (u, v) ∉ E thì tuỳ theo trường hợp cụ thể, C[u, v] được gán một giá trị nào đó để có thể nhận biết được (u, v) không phải là cạnh (Chẳng hạn có thể gán bằng +∞, hay bằng 0, bằng -∞, v.v…)
- Quy ước c[v, v] = 0 với mọi đỉnh v.

Đường đi, chu trình trong đồ thị có trọng số cũng được định nghĩa giống như trong trường hợp không trọng số, chỉ có khác là độ dài đường đi không phải tính bằng số cạnh đi qua, mà được tính bằng tổng trọng số của các cạnh đi qua.

Input: file văn bản MINPATH.INP

- Dòng 1: Chứa số đỉnh n (≤ 1000), số cung m của đồ thị, đỉnh xuất phát s, đỉnh đích f cách nhau ít nhất 1 dấu cách
- m dòng tiếp theo, mỗi dòng có dạng ba số u, v, c[u, v] cách nhau ít nhất 1 dấu cách, thể hiện (u, v) là một cung ∈ E và trọng số của cung đó là c[u, v] (c[u, v] là số nguyên có giá trị tuyệt đối ≤ 1000)

Output: file văn bản MINPATH.OUT ghi đường đi ngắn nhất từ s tới f và độ dài đường đi đó



MINPATH.INP	MINPATH.OUT
6 7 1 4 1 2 1 1 6 20 2 3 2 3 6 3 3 4 20 5 4 5 6 5 4	Distance from 1 to 4: 15 4<-5<-6<-3<-2<-1

Viết hàm Load đồ thị thì file input và hàm in kết quả ra file output.

Bài 5 (3 điểm): Viết thuật toán tìm đường đi ngắn nhất theo gợi ý sau.

```
for (∀v ∈ V) do d[v] := +∞;
d[s] := 0;
repeat
  Stop := True;
  for (∀u ∈ V) do
    for (∀v ∈ V: (u, v) ∈ E) do
      if d[v] > d[u] + c[u, v] then
        begin
          d[v] := d[u] + c[u, v];
          Stop := False;
        end;
until Stop;
```

```
procedure Ford_Bellman; {Thuật toán Ford-Bellman}
var
  Stop: Boolean;
  u, v, CountLoop: Integer;
begin
  for CountLoop := 1 to n - 1 do
    begin
      Stop := True;
      for u := 1 to n do
        for v := 1 to n do
          if d[v] > d[u] + c[u, v] then  {Nếu ∃u, v thoả mãn d[v] > d[u] + c[u, v] thì tối ưu lại d[v]}
            begin
              d[v] := d[u] + c[u, v];
              Trace[v] := u; {Lưu vết đường đi}
              Stop := False;
            end;
      if Stop then Break;
    end;
  {Thuật toán kết thúc khi không sửa nhãn các d[v] được nữa hoặc đã lặp đủ n - 1 lần}
end;
```