[Thuật toán] Tìm đường đi ngắn nhất Dijkstra, Floyd

October 13, 2013 / CLB Tin học ICTU / Thuật toán Đồ thị

Update 25/05/2014: Do một số góp ý của các bạn nên mình đã viết thêm 1 chương trình của thuật toán Dijkstra theo cấu trúc hàm và cũng nhân tiện chỉnh lại chút code cho sáng sủa và chính xác hơn ^^.

Update 27/09/2014: bổ xung code pascal của thuật toán tại đây: http://ideone.com/c7J0dq

Update 14/06/2014: Chương trình mô phỏng thuật toán Dijkstra

Nội dung

Trong bài viết này chỉ đề cập tới các thuật toán tìm đường đi ngắn nhất Dijkstra và Floyd, một số thuật ngữ liên quan mình sẽ không giải thích hay định nghĩa, các bạn tự tìm hiểu trong sách hoặc trên mạng.

Bài toán đường đi ngắn nhất nguồn đơn là bài toán tìm một đường đi giữa hai đỉnh sao cho tổng các trọng số của các cạnh tạo nên đường đi đó là nhỏ nhất. Hay nói một cách toán học là:
Cho đơn đồ thị liên thông, có trọng số G=(V,E). Tìm khoảng cách d(a,b) từ một đỉnh a cho trước đến một đỉnh b bất kỳ của G và tìm đường đi ngắn nhất từ a đến b.

Như tiêu đề bài viết, chúng ta sẽ tìm hiểu 2 thuật toán để giải quyết bằng cách sử dụng mà trận kề cảu đồ thị(chú ý ta xét trọng số của đồ thị là không âm).

Ma trận kề của đồ thị có n đỉnh là ma trận vuông G có số hàng số cột là n. G[i][j] là độ dài đường đi từ đỉnh i tới đỉnh j. Nếu xét đồ thị vô hướng thì G[i][j] = G[j][i]. Độ dài từ một đỉnh tới chính nó luôn là 0 (G[i][i] = 0). Nếu giữa 2 cạnh i và j của đồ thị không tồn tại đường đi thì G[i][j] = vô cùng (∞). Tuy nhiên khi biểu diễn trong máy tính thì giá trị ∞ được đặt là 1 hằng số rất lớn hoặc là tổng các giá trị trong ma trận (tổng độ dài các cạnh).

1. Thuật toán Dijkstra

Về thuật toán Dijkstra có 2 loại là tìm đường đi ngắn nhất từ 1 đỉnh nguồn tới 1 đỉnh đích và tìm đường đi ngắn nhất từ 1 đỉnh nguồn tới các đỉnh còn lại của đồ thị, và ở đây mình sẽ nói về loại thứ 1. (loại thứ hai bạn có thể tìm trên mạng hoặc chỉ cần thay đổi dòng **while (s[b] == 0) (dòng 43 của code 1 & dòng 76 của code 2) thành vòng for duyệt từ 0 đến n-1 là sẽ tìm được tất cả các đỉnh).**

– Dùng 1 mảng Len[] – Len[i] là khoảng cách ngắn nhất từ đỉnh a tới đỉnh i.

– Dùng 1 mảng S đánh dấu các đỉnh i đặc biệt (các đỉnh i mà thời điểm hiện tại thì đường đi từ a tới i là ngắn nhất).

– Dùng mảng P[] đánh dấu đường đi. P[j] = i nếu i là đỉnh đi trước j trong đường đi ngắn nhất.

– Đặt lại giá trị vô cùng cho các cặp đỉnh không có đường đi.

– Khởi tạo tất cả các đường đi từ a đến các đỉnh khác bằng vô cùng.

– Khởi tạo đường đi từ a đến chính a = 0.

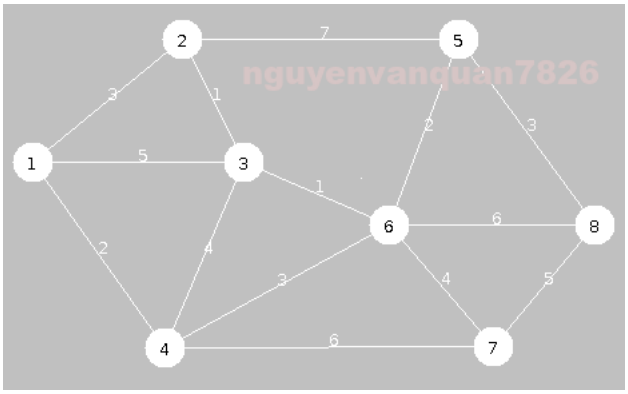
– Duyệt hết các đỉnh V của đồ thị

+ Tìm đỉnh i chưa nằm trong S mà đường đi từ a tới i là ngắn nhất để đưa vào S. Nếu không tìm được đỉnh nào nghĩa là đã duyệt hết các đỉnh có thể đi mà vẫn chưa thấy đỉnh đích => không thể đi được.

+ Nếu tìm được đỉnh i thì duyệt tất cả các đỉnh j chưa nằm trong S. Nếu Len[i] + G[i][j] < Len[j] (trong đó G[i][j] là khoảng cách từ đỉnh i tới đỉnh j) thì gán Len[j] = Len[i] + G[i][j]; và đánh dấu đường đi P[j] = i.

Lưu ý: Do trong C, mảng bắt đầu từ 0. Do vậy các đỉnh khi tính toán thì sẽ tính từ đỉnh 0 đến đỉnh n-1. Tuy nhiên khi hiển thị ra thì vẫn phải là từ đỉnh 1 đến n và trong file input.inp thì đỉnh đầu và đỉnh cuối cũng sẽ được tính từ 1 đến n. Do đó trong code trước khi tính toán ta cần giảm đỉnh đầu và đỉnh cuối đi 1 đơn vị. Sau khi tính toán xong thì khi xuất kết quả lại cần tăng các đỉnh trong đường đi tìm được lên 1 đơn vị để hiển thị đúng (VD ta muốn tính đường đi từ đỉnh 4 đến đỉnh 8, thì đỉnh 4 tương ứng với vị trí thứ 3 trong mảng, đỉnh 8 ứng với vị trí thứ 7 nên ta cần giảm 4 xuống 3, 8 xuống 7 để tính toán. Khi tìm được đường đi, giả sử là 3 -> 5 -> 4 -> 7 thì phải in ra là 4 -> 6 -> 5 -> 8).

Chúng ta sẽ đi thực hành với đồ thị sau theo 2 code là làm ngay trong main và thực hiện theo các hàm:



file **input.inp**: hàng đầu tiên thể hiện có 8 điểm, đi từ điểm 4 đến điểm 8. ma trận 8×8 ở dưới là ma trận kề của đồ thị.

[code]
8 4 8
0 3 5 2 0 0 0 0
3 0 1 0 7 0 0 0
5 1 0 4 0 1 0 0
2 0 4 0 0 3 6 0
0 7 0 0 0 2 0 3
0 0 1 3 2 0 4 6
0 0 0 6 0 4 0 5
0 0 0 0 3 6 5 0
[/code]

*** Code ngay trong main**
Code này đã được sửa và khắc phục một số lỗi từ code ngày trước (hoặc link dự phòng).

[code language="cpp"]
#include <stdio.h>
#include <stdlib.h>
#define INP "input.inp"
#define OUT "output.out"

int main() {
FILE *fi = fopen(INP, "r");
FILE *fo = fopen(OUT, "w");
int n, a, b, i, sum = 0;

// nhap du lieu tu file input
fscanf(fi, "%d%d%d", &n, &a, &b);

```
int G[n][n];
int S[n], Len[n], P[n];

// nhap ma tran va tinh gia tri vo cung (sum)
for (i = 0; i < n; i++)
for (int j = 0; j < n; j++) {
fscanf(fi, "%d", &G[i][j]);
sum += G[i][j];
}
// dat vo cung cho tat ca cap canh khong noi voi nhau
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
if (i != j && G[i][j] == 0)
G[i][j] = sum;
}
}

/* Do mang tinh tu G[0][0] nen can giam vi tri
di 1 don vi de tinh toan cho phu hop*/
a–;
b–;

for (int i = 0; i < n; i++) {
Len[i] = sum; // khoi tao do dai tu a toi moi dinh la vo cung
S[i] = 0; // danh sach cac diem da xet
P[i] = a; // dat diem bat dau cua moi diem la a
}

Len[a] = 0; // dat do dai tu a -> a la 0

// tim duong di ngan nhat tu 1 dinh den moi dinh khac thi thay bang vong for:
//for (int k = 0; k < n; k++)
while (S[b] == 0) { // trong khi diem cuoi chua duoc xet
for (i = 0; i < n; i++) // tim 1 vi tri ma khong phai la vo cung
if (!S[i] && Len[i] < sum)
break;

// i >=n tuc la duyet het cac dinh ma khong the tim thay dinh b -> thoat
if (i >= n) {
printf("done dijkstra\n");
break;
}

for (int j = 0; j < n; j++) { // tim diem co vi tri ma do dai la min
if (!S[j] && Len[i] > Len[j]) {
i = j;
}
}

S[i] = 1; // cho i vao danh sach xet roi

for (int j = 0; j < n; j++) { // tinh lai do dai cua cac diem chua xet
if (!S[j] && Len[i] + G[i][j] < Len[j]) {
Len[j] = Len[i] + G[i][j]; // thay doi len
P[j] = i; // danh dau diem truoc no
}
}
}
```

```
printf("done dijkstra\n");

/* Do ta dang tinh toan tu dinh 0 nen
muon hien thi tu dinh 1 thi can dung i + 1 de phu hop */

printf("start find path\n");

if (Len[b] > 0 && Len[b] < sum) {
fprintf(fo, "Length of %d to %d is %d\n", a + 1, b + 1, Len[b]);

// truy vet
while (i != a) {
fprintf(fo, "%d <– ", i + 1);
i = P[i];
}
fprintf(fo, "%d", a + 1);
} else {
fprintf(fo, "khong co duong di tu %d den %d\n", a + 1, b + 1);
}

printf("done find path\n");

fclose(fi);
fclose(fo);

printf("done – open file output to see result\n");
return 0;
}
```

[/code]

link dự phòng

*** Code theo từng hàm**

Trong code theo hàm có hàm:

– **readData** thực hiện đọc thông tin từ file input.

– **dijkstra** thực hiện thuật toán

– **back** thực hiện trả về chuỗi là đường đi tìm được

– **outResult** thực hiện in ra file output kết quả

```
[code language="cpp"]
#include <stdio.h>
#include <stdlib.h>
#include <cstring>

#define INP "input.inp"
#define OUT "output.out"

// read data in file input
int readData(int ***G, int *n, int *a, int *b) {
FILE *fi = fopen(INP, "r");
if (fi == NULL) {
printf("file input not found!\n");
return 0;
}
printf("start read file\n");

fscanf(fi, "%d %d %d", n, a, b);

*G = (int **) malloc((*n) * sizeof(int));
for (int i = 0; i < *n; i++) {
```

```
(*G)[i] = (int *) malloc((*n) * sizeof(int));
for (int j = 0; j < *n; j++) {
int x;
fscanf(fi, "%d", &x);
(*G)[i][j] = x;
}
}

fclose(fi);
printf("done read file\n");
return 1;
}

// thuat toan dijkstra
int dijkstra(int **G, int n, int a, int b, int P[]) {

/* Do mang tinh tu G[0][0] nen can giam vi tri
di 1 don vi de tinh toan cho phu hop*/
a−;
b−;

printf("start dijkstra\n");

int* Len = (int *) malloc(n * sizeof(int));
int* S = (int *) malloc(n * sizeof(int));

int sum = 0; // gia tri vo cung

// tinh gia tri vo cung (sum)
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
sum += G[i][j];
}
}

// dat vo cung cho tat ca cap canh khong noi voi nhau
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
if (i != j && G[i][j] == 0)
G[i][j] = sum;
}
}

for (int i = 0; i < n; i++) {
Len[i] = sum; // khoi tao do dai tu a toi moi dinh la vo cung
S[i] = 0; // danh sach cac diem da xet
P[i] = a; // dat diem bat dau cua moi diem la a
}

Len[a] = 0; // dat do dai tu a -> a la 0

int i;

// tim duong di ngan nhat tu 1 dinh den moi dinh khac thi thay bang vong for:
//for (int k = 0; k < n; k++)
while (S[b] == 0) { // trong khi diem cuoi chua duoc xet
for (i = 0; i < n; i++) // tim 1 vi tri ma khong phai la vo cung
if (!S[i] && Len[i] < sum)
break;
```

```
// i >=n tuc la duyet het cac dinh ma khong the tim thay dinh b -> thoat
if (i >= n) {
printf("done dijkstra\n");
return 0;
}

for (int j = 0; j < n; j++) { // tim diem co vi tri ma do dai la min
if (!S[j] && Len[i] > Len[j])
i = j;
}

S[i] = 1; // cho i vao danh sach xet roi

for (int j = 0; j < n; j++) { // tinh lai do dai cua cac diem chua xet
if (!S[j] && Len[i] + G[i][j] < Len[j]) {
Len[j] = Len[i] + G[i][j]; // thay doi len
P[j] = i; // danh dau diem truoc no
}
}
}
printf("done dijkstra\n");
return Len[b];
}

// truy vet duong di
void back(int a, int b, int *P, int n, char *path) {

//char *path = (char *) malloc((n * 10) * sizeof(char));

/* Do mang tinh tu G[0][0] nen can giam vi tri
di 1 don vi de tinh toan cho phu hop*/
a--;
b--;

printf("start find path\n");

int i = b;
int point[n]; // danh sach cac dinh cua duong di
int count = 0;

/* Do ta dang tinh toan tu dinh 0 nen
muon hien thi tu dinh 1 thi can dung i + 1 de phu hop */

point[count++] = i + 1;
while (i != a) {
i = P[i];
point[count++] = i + 1;
}

strcpy(path, "");
char temp[10];
for (i = count - 1; i >= 0; i--) {
sprintf(temp, "%d", point[i]);
strcat(path, temp);

if (i > 0) {
sprintf(temp, " --> ");
strcat(path, temp);
```

```
}
}

printf("done find path\n");
}

void outResult(int len, char* path) {
FILE *fo = fopen(OUT, "w");

if (len > 0) {
fprintf(fo, "\nLength of %c to %c is %d\n", path[0],
path[strlen(path) – 1], len);
}

fprintf(fo, "path: %s\n", path);

fclose(fo);
}

int main() {
int **G, n, a, b, len;

if (readData(&G, &n, &a, &b) == 0) {
return 0;
}

char *path = (char *) malloc((10 * n) * sizeof(char));
int P[n];

len = dijkstra(G, n, a, b, P);

if (len > 0) {
back(a, b, P, n, path);
outResult(len, path);
} else {
char *path = (char *) malloc((n * 10) * sizeof(char));
sprintf(path, "khong co duong di tu %d den %d\n", a, b);
outResult(len, path);
}

printf("done – open file output to see result\n");
return 0;
}
```

[/code]

link dự phòng

Nhìn code có vẻ hơi dài nhưng khi đọc hiểu rồi thì chả dài tẹo nào @@ =)).

2. Thuật toán Floyd

Thuật toán này cho phép chúng ta tìm đường đi ngắn nhất giữa mọi cặp đỉnh.

Nếu đỉnh k nằm trên đường đi ngắn nhất từ đỉnh i tới đỉnh j thì đoạn đường từ i tới k và từ k tới j phải là đường đi ngắn nhất từ i tới k và từ k tới j tương ứng. Do đó ta sử dụng ma trận A để lưu độ dài đường đi ngắn nhất giữa mọi cặp đỉnh.
– Ban đầu ta đặt A[i,j] = C[i,j], tức là ban đầu A chứa độ dài đường đi trực tiếp (không đi qua đỉnh nào cả).
– Sau đó thực hiện n lần lặp, sau lần lặp thứ k, ma trận A sẽ chứa độ dài đường đi ngắn nhất giữa mọi cặp đỉnh chỉ đi qua các đỉnh thuộc tập {1,2,..,k}. Như vậy, sau n lần lặp ta nhận được ma trận A chứa độ dài các đường đi ngắn nhất giữa mọi cặp đỉnh của đồ thị.

– Ký hiệu Ak là ma trận A sau lần lặp thứ k, tức là Ak[i,j] là độ dài đường đi ngắn nhất từ i đến j chỉ đi qua các đỉnh thuộc {1, 2,.., k}. Ak[i,j] được tính theo công thức như sau: Ak[i,j] = min {Ak -1[i,j], Ak-1[i,k] + Ak-1[k,j]}.

– Trong quá trình lặp ta phải lưu lại vết đường đi, tức là đường đi ngắn nhất đi qua các đỉnh nào. Khi đó ta sử dụng mảng phụ P[nxn], trong đó P[i,j] lưu đỉnh k nếu đường đi ngắn nhất từ i đến j đi qua đỉnh k. Ban đầu P[i,j]=0 với mọi i,j, vì lúc đó đường đi ngắn nhất là đường đi trực tiếp, không đi qua đỉnh nào cả.

Code thuật toán:

```
[code language="cpp"]
void Floyd (int a, int b)
{
int max = tongthiethai();
for (int i=0; i<n; i++)
for (int j=0; j<n; j++)
{
if (G[i][j])
A[i][j] = G[i][j];
else A[i][j] = max;
P[i][j] = -1;
}

for (int k=0; k<n; k++) // lap n lan
{
for (int i=0; i<n; i++) // thay doi do dai duong di cua cac dinh
for (int j=0; j<n; j++)
if (A[i][j] > A[i][k] + A[k][j])
{
A[i][j] = A[i][k] + A[k][j];
P[i][j] = k ;
}
}
}
[/code]
```

Cách xây dựng chương trình hoàn chỉnh hoàn toàn giống với thuật toán Dijkstra.

Code nâng cao

Đây là code cho phép chọn 1 trong 2 thuật toán trên và xuất ra file đúng theo quá trình làm như các kết quả trong hình bên dưới hoặc link dự phòng

file input.inp:

```
8
A B C D E F G H
0 3 5 2 0 0 0 0
3 0 1 0 7 0 0 0
5 1 0 4 0 1 0 0
2 0 4 0 0 3 6 0
0 7 0 0 0 2 0 3
0 0 1 3 2 0 4 6
0 0 0 6 0 4 0 5
0 0 0 0 3 6 5 0
```

Menu console

```
Ma tran ke cua do thi

 A B C D E F G H

 0 3 5 2 0 0 0 0
 3 0 1 0 7 0 0 0
 5 1 0 4 0 1 0 0
 2 0 4 0 0 3 6 0
 0 7 0 0 0 2 0 3
 0 0 1 3 2 0 4 6
 0 0 0 6 0 4 0 5
 0 0 0 0 3 6 5 0

Moi ban chon thuat toan :
1: Thuat toan Dijkstra
2: Thuat toan Floyd
3: Thoat !
1

-----Thuat toan Dijkstra-----

Cac dinh danh so tu 1 den 8.Hoac tu A den H
Nhap dinh bat dau : 2
Nhap dinh ket thuc : 6
Hoan thanh ! Mo file output.out de xem ket qua !Press any key to continue . . .


Moi ban chon thuat toan :
1: Thuat toan Dijkstra
2: Thuat toan Floyd
3: Thoat !
2

-----Thuat toan Floy-----

Cac dinh danh so tu 1 den 8.Hoac tu A den H
Nhap dinh bat dau : 3
Nhap dinh ket thuc : 5
Hoan thanh ! Mo file output.out de xem ket qua !Press any key to continue . . .


Moi ban chon thuat toan :
1: Thuat toan Dijkstra
2: Thuat toan Floyd
3: Thoat !
```


Output Dijkstra

```
Ma tran ke cua do thi

 A B C D E F G H

 0 3 5 2 0 0 0 0
 3 0 1 0 7 0 0 0
 5 1 0 4 0 1 0 0
 2 0 4 0 0 3 6 0
 0 7 0 0 0 2 0 3
 0 0 1 3 2 0 4 6
 0 0 0 6 0 4 0 5
 0 0 0 0 3 6 5 0
```

Thuật toán Dijkstra

*****

TT	1(A)	2(B)	3(C)	4(D)	5(E)	6(F)	7(G)	8(H)
1	[~,~]	*[0,2]	[~,~]	[~,~]	[~,~]	[~,~]	[~,~]	[~,~]
2	[3,2]	-	*[1,2]	[~,2]	[7,2]	[~,2]	[~,2]	[~,2]
3	[3,2]	-	-	[5,3]	[7,2]	*[2,3]	[~,2]	[~,2]
4	*[3,2]	-	-	[5,3]	[4,6]	-	[6,6]	[8,6]
5	-	-	-	[5,3]	*[4,6]	-	[6,6]	[8,6]
6	-	-	-	*[5,3]	-	-	[6,6]	[7,5]
7	-	-	-	-	-	-	*[6,6]	[7,5]
8	-	-	-	-	-	-	-	*[7,5]

```
Do dai ngan nhat cua duong di tu 2(B) den 6(F) la 2
Qua trinh duong di: 2 --> 3 --> 6
```

Output Floyd

```
Buoc thu 6
                                Thuật toán Floyd
                A                                       P
   4   3   4   2   7   5   8  10      4   0   2   0   6   3   4   6
   3   2   1   5   4   2   6   7      0   3   0   1   6   3   6   6
   4   1   2   4   3   1   5   6      2   0   2   0   6   0   6   6
   2   5   4   4   5   3   6   8      0   1   0   1   6   0   0   6
   7   4   3   5   4   2   6   3      6   6   6   6   6   0   6   0
   5   2   1   3   2   2   4   5      3   3   0   0   0   3   0   5
   8   6   5   6   6   4   8   5      4   6   6   0   6   0   6   0
  10   7   6   8   3   5   5   6      6   6   6   6   0   5   0   5

Buoc thu 7                      nguyenvanquan7826
                A                                       P
   4   3   4   2   7   5   8  10      4   0   2   0   6   3   4   6
   3   2   1   5   4   2   6   7      0   3   0   1   6   3   6   6
   4   1   2   4   3   1   5   6      2   0   2   0   6   0   6   6
   2   5   4   4   5   3   6   8      0   1   0   1   6   0   0   6
   7   4   3   5   4   2   6   3      6   6   6   6   6   0   6   0
   5   2   1   3   2   2   4   5      3   3   0   0   0   3   0   5
   8   6   5   6   6   4   8   5      4   6   6   0   6   0   6   0
  10   7   6   8   3   5   5   6      6   6   6   6   0   5   0   5

Do dai ngan nhat cua duong di tu 7(G) den 5(E) la 6
Qua trinh duong di: 7 --> 6 --> 5
```

Dijkstra | đường đi ngắn nhất | Floyd | thuật toán Dijkstra | thuật toán Floyd | thuật toán tìm đường đi ngắn nhất

[Assembly] Kết hợp Assembly với ngôn ngữ bậc cao

[WordPress] Đánh công thức toán trên wordpress không dùng MathType

100 thoughts on "[Thuật toán] Tìm đường đi ngắn nhất Dijkstra, Floyd"

Pingback: [Tự sướng] Kỷ niệm Blog tròn 1 tuổi | Chia sẻ để cuộc sống tốt đẹp hơn!

Thái says:
June 13, 2014 at 5:45 pm

cho em hoi sao em chay moi do thi deu khong tim duoc duong di vay. nhung giai tay thi van co.

Reply

nguyenvanquan7826 says:
June 13, 2014 at 10:35 pm

@@ Bạn xem kỹ lại code với file input nhé. Mình chạy ok.

Reply

Rias Gremory says:
May 11, 2015 at 8:30 am

Bạn ơi sao copy code full cuối bài về khi chạy bị lỗi này
[Error] C:UsersAdminDownloadsideone_DpaGPW.cpp:167: E2313 Constant expression required in function Dijkstra(GRAPH,int,int)

[Error] C:UsersAdminDownloadsideone_DpaGPW.cpp:283: E2313 Constant expression required in function floyd(GRAPH,int,int)

Reply

nguyenvanquan7826 says:
May 11, 2015 at 9:21 am

Bạn copy code nào vậy? gửi lại file cho mình vào mail nguyenvanquan7826@gmail.com mình xem cho nhé.

Reply

Rias Gremory says:
May 12, 2015 at 2:35 pm

minh gởi mail rồi đó

nguyenvanquan7826 says:
May 12, 2015 at 3:55 pm

Mình nhận được rồi, mình lấy code đó chạy vẫn ổn, không vấn đề gì. Có lẽ bạn dùng C-free nên nó không được chuẩn lắm hoặc lỗi gì đó. Bạn bật teamview mình xem cho nhé. Liên lạc với mình qua skye: nguyenvanquan7826

Pingback: [Java - Thuật toán] Mô phỏng thuật toán Dijkstra | Chia sẻ để cuộc sống tốt đẹp hơn!

Pingback: [Java - Thuật toán] Mô phỏng thuật toán Dijkstra tìm đường đi ngắn nhất | Chia sẻ để cuộc sống tốt đẹp hơn!

itmietvuon says:
September 23, 2014 at 2:51 pm

Bạn có thể đổi code sang Pascal được không vậy? Biết 99% câu trả lời sẽ là "Mình đang bận lắm!" nhưng vẫn hy vọng bạn sẽ chuyển thành 1 bản code bằng Pascal để phổ biến kiến thức nhiều hơn đến các newbie không rành về C.
Cám ơn bạn đã đọc.

Reply

nguyenvanquan7826 says:
September 23, 2014 at 11:41 pm

Mình sẽ cố gắng chuyển đổi code sang pascal trong tuần này nhé. Cảm ơn bạn đã quan tâm tới các bài viết trong blog và đặc biệt là tinh thần chia sẻ của bạn 🙂

Reply

nguyenvanquan7826 says:
September 27, 2014 at 2:35 am

Bạn có thể xem code pascal tại đây nhé http://ideone.com/c7J0dq

Reply

itmietvuon says:
September 28, 2014 at 4:36 am

Cám ơn bạn rất nhiều, mong VN sẽ ngày mỗi nhiều hơn những người như bạn.
Chúc bạn sớm có thêm nhiều bài viết hay cho cộng đồng những người yêu lập trình.

Reply

nguyenvanquan7826 says:
September 28, 2014 at 11:37 pm

Cảm ơn bạn 🙂 Mình sẽ cố gắng viết tốt hơn 😉

Reply

Chile says:
November 25, 2014 at 7:57 am

Anh ơi giúp em bài này với! Em đang làm tiểu luận môn học, đề bài của em như sau:
Đồ thị vô hướng G={V,E} được cho bởi danh sách cạnh DS. Cho u, v thuộc V. Xây dựng thuật toán tìm hai đường đi A1(u,v) và A2(u,v) sao cho không có cạnh nào chung và có tổng độ dài ngắn nhất.
Em loay hoay mãi mà không biết bắt đầu từ đâu nữa ạ. Mong anh giúp dùm em.

Reply

nguyenvanquan7826 says:
November 25, 2014 at 3:25 pm

Cái này bản chất là tìm đường đi ngắn nhất mà.
A1 + A2 ngắn nhất khi A1 ngắn nhất và A2 ngắn nhất. Bạn dùng dijkstra tìm A1, A2, trong quá trình tìm A2 thì mỗi lần tìm được 1 cạnh phải xem cạnh đó có trong A1 chưa nhé.

Reply

Chile says:
November 26, 2014 at 4:27 pm

thanks anh ạ

Reply

bd says:
November 26, 2014 at 12:00 pm

thuật toán này anh đã làm vs Heap chưa ạ? Em ko rõ Dijkstra Heap lắm. Nếu chưa, anh có thể làm hay chỉ cho e đc k ạ?

Reply

nguyenvanquan7826 says:
November 26, 2014 at 3:15 pm

Cái này anh chưa làm với Heap. Cũng chưa nghịch Heap bao giờ lun :D. Em thử search trên mạng xem có không 😀

Reply

bd says:
November 26, 2014 at 4:34 pm

vâng ạ. Anh cho em hỏi thêm. Với loại 2, em muốn tìm từ 1 đỉnh tới các định thì ngoài thay thành vòng for thì em còn phải đổi những j nữa ạ, như phần input em cũng k biết phải ghi sao. Ngoài ra, em ko chạy đc code theo từng phần của anh ạ. Nó chạy tới phần start dijkstra thì crush ạ. Anh có thể chỉ cho e đc k ạ?

Reply

nguyenvanquan7826 says:
November 26, 2014 at 6:16 pm

Khi chạy từ 1 đỉnh tới mọi đỉnh khác thì chỉ thay vòng for duyệt hết toàn bộ các đỉnh là xong, không cần thêm gì khác. Code theo từng hàm thì bạn cứ xây dựng bình thường là được mà.

Reply

bd says:
November 26, 2014 at 8:50 pm

vậng em cảm ơn

huyền says:
November 26, 2014 at 9:44 pm

bạn ơi cho mình hỏi tí..mình có làm đồ án về cái này mà dùng java bạn có thể giải thích rõ hơn về code phần java của bạn đc ko

Reply

huyền says:
November 26, 2014 at 9:45 pm

quên,,mình cần lắm có gì bạn giúp mình với nhá..mình cảm ơn bạn nhiều

Reply

nguyenvanquan7826 says:
November 26, 2014 at 11:28 pm

Bạn cứ làm đi, nếu được mình sẽ giúp :v

Reply

huyền says:
November 27, 2014 at 11:43 am

bạn có thể giải thích dùm mình code phần chạy dãy thuật đc ko :((.bạn cho mình sđt bạn đi

Reply

nguyenvanquan7826 says:
November 27, 2014 at 12:00 pm

Chạy dãy thuật là sao bạn?
có gì bạn có thể gọi vào số của mình.
096.567.7826

Reply

huyền says:
November 27, 2014 at 11:45 am

thấy bạn trl cmt mà rơi nước mắt vì mừng 🙂

Reply

nguyenvanquan7826 says:
November 27, 2014 at 12:00 pm

Làm gì đến mức đó 😀

Reply

doveandrose says:
December 1, 2014 at 5:50 pm

chào bạn Nguyễn Văn Quân , mình có thể mời bạn cho biết đáp số của bài toán sau đây dc ko ?
cho hệ trục tọa độ 0xy , có các điểm sau đây
A1(586,363),A2(254,137),A3(467,516),A4(798,472),A5(599,213),A6(372,344),A7(146,412),
A8(818,346),A9(850,199),A10(314,260),A11(72,268),A12(731,57),A13(429,179),A14(499,81),
A15(89,169),A16(113,82),A17(904,59),A18(926,551),A19(54,20)
Giả sử rằng 2 điểm bất kỳ đều có đoạn nối và vô hướng
Độ dài đoạn nối (kiểu số thực) giữa 2 điểm bất kỳ dc tính theo kiểu toán phổ thông đã học
Điểm bắt đầu : A18(926,551)
Cho biết độ dài đường đi (kiểu số thực) ngắn nhất xuất phát từ A18 và đi qua tất cả các đỉnh còn lại (mỗi đỉnh phải đi qua đúng 1 lần)

Reply

nguyenvanquan7826 says:
December 1, 2014 at 10:44 pm

Với bài này nó lại là một thuật toán khác rồi bạn. Mình sẽ cố gắng xem cách giải nó thế nào.

Reply

myth says:
January 12, 2015 at 5:57 pm

thật sự mình quá lơ mơ về thuật toán của bạn k hiểu được nhiều

Reply

nguyenvanquan7826 says:
January 12, 2015 at 8:42 pm

Bạn không hiểu code hay thuật toán? Không hiểu thuật dijkstra hay floyd.

Reply

thao says:
January 15, 2015 at 4:29 pm

mình làm thực tập cơ sở vè thuật toán này,,,thì giao diện ra nó sẽ như thế nào?

Reply

nguyenvanquan7826 says:
January 15, 2015 at 5:53 pm

Cái này là tùy bạn thôi. Bạn có thể tham khảo chương trình của mình tại đây:
http://nguyenvanquan7826.com/2014/06/14/java-thuat-toan-mo-phong-thuat-toan-dijkstra-tim-duong-di-ngan-nhat/

Reply

Phuong says:
January 15, 2015 at 6:44 pm

anh ơi có code java ko vậy? C e chẳng biết gì cả

Reply

nguyenvanquan7826 says:
January 15, 2015 at 8:48 pm

Học đk java mà không rõ C? nó như nhau, chả khác gì luôn.

Reply

caocuong says:
January 30, 2015 at 1:34 am

ad ơi, e đang có vấn đề về : tìm đường đi dài nhất trên đồ thị k có chu trình bằng thuật toán PERT .. ad giúp e đc k?

Reply

nguyenvanquan7826 says:
January 30, 2015 at 10:03 am

Cái này anh chưa làm mà giờ cũng chưa có thời gian làm nữa 🙂 Em thông cảm nhá.

Reply

dattrinh says:
February 3, 2015 at 2:03 pm

bạn ơi mình sửa 1 tí ở code của bạn là nhập 2 điểm từ bàn phím và chuyển scanf printf thành cin và cout nhưng ko chạy được, bạn giúp mình tí nhé

```
#include
#include
#include
#define INP "input.txt"
using namespace std;

int main() {
FILE *fi = fopen("input.txt", "r");

int n, a, b, i, sum = 0;
cout<>a;
cout<>b;
fscanf(fi, "%d%d%d", &n);
int G[n][n];
int S[n], Len[n], P[n];
for (i = 0; i < n; i++)
for (int j = 0; j < n; j++) {
fscanf(fi, "%d", &G[i][j]);
sum += G[i][j];
}

for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
if (i != j && G[i][j] == 0)
G[i][j] = sum;
}
}
a--;
b--;

for (int i = 0; i < n; i++) {
Len[i] = sum;
```

```
S[i] = 0;
P[i] = a;
}

Len[a] = 0;
while (S[b] == 0) {
for (i = 0; i < n; i++)
if (!S[i] && Len[i] < sum)
break;
for (int j = 0; j Len[j]) {
i = j;
}
}

S[i] = 1;

for (int j = 0; j < n; j++) {
if (!S[j] && Len[i] + G[i][j] 0 && Len[b] < sum) {
cout<< " duong di ngan nhat tu dinh "<<a + 1<<" den dinh"<<b + 1<<" la "<< Len[b];

while (i != a) {
cout<< i + 1<<"<– ";
i = P[i];
}
cout<< a + 1;
} else {
cout<<"khong co duong di tu "<<a + 1<<" den "<< b + 1;
}
fclose(fi);

return 0;
}
```

Reply

nguyenvanquan7826 says:
February 3, 2015 at 4:50 pm

Bạn xem lại 2 dòng này nhá.
cout<>a;
cout<>b;

cin là dùng >>
cout là dùng <<

Reply

dattrinh says:
February 3, 2015 at 5:56 pm

mình dùng cin>> và cout << đó bạn nhưng ko được

Reply

dattrinh says:
February 3, 2015 at 5:58 pm

mình chỉ chạy được nhập a và b thôi sau đó chương trình sẽ ko chạy nữa nó áo là stop working

Reply

dattrinh says:
February 3, 2015 at 6:01 pm

mình chạy thì nó chỉ nhập a và b xong rồi thì nó báo stop working

Reply

nguyenvanquan7826 says:
February 4, 2015 at 12:25 am

Ở trên mình thấy bạn viết thế này mà

cout<>a;
cout<>b;
fscanf(fi, "%d%d%d", &n);

cout viết <>
khi nhập mỗi n thì chỉ có 1 cái %d thôi chứ, 3 cái lận…

Reply

raven says:
March 10, 2015 at 9:51 am

bạn ơi cho mình hỏi ở dòng
" for (int j = 0; j Len[j])
i = j;
thì !S[j] có ý nghĩa gì?

Reply

nguyenvanquan7826 says:
March 11, 2015 at 5:35 pm

!S[j] tức là điểm j chưa được xét đó.

Reply

Mac says:
March 12, 2015 at 8:35 am

tại sao bạn lại xuất đường đi là b<-- ... <--a sao không xuất ngược lại

Reply

nguyenvanquan7826 says:
March 12, 2015 at 11:34 am

Tại vì thứ tự đỉnh tìm được nó ngược nên cần làm thế. 🙂

Reply

Mac says:
March 14, 2015 at 4:07 pm

mình có thể đổi lại được không

nguyenvanquan7826 says:
March 15, 2015 at 1:49 pm

Được, bạn xem phần code theo hàm ở bên trên nhé.

Nevermore says:
March 14, 2015 at 4:52 pm

Anh ơi, em lấy code hoàn chỉnh của anh ở http://pastebin.com/FiZzb3UH copy vao devc++ chạy thì nó không ra giao diện như của anh ạ, nó chạy như ma trận ấy. Em dùng Devc++5.7.1 ạ. Anh chỉ em làm sai chỗ nào với!

Reply

nguyenvanquan7826 says:
March 15, 2015 at 1:43 pm

Là sao bạn, cái code này chạy nó sẽ ra giao diện như mấy hình phía cuối bài viết của mình mà? Bạn muốn nó hiện như thế nào?

Reply

thuylanh says:
April 14, 2015 at 4:37 pm

a có thể chuyển code này sang java ko ạ.

Reply

nguyenvanquan7826 says:
April 14, 2015 at 11:44 pm

C với java khác gì đâu bạn. Nếu muốn có thể xem bài này nhé:
http://nguyenvanquan7826.com/2014/06/14/java-thuat-toan-mo-phong-thuat-toan-dijkstra-tim-duong-di-ngan-nhat/

Reply

VuDuc says:
April 19, 2015 at 7:53 pm

b ơi! b có thể giúp mình làm với đồ thị có hướng dc k????

Reply

VuDuc says:
April 19, 2015 at 8:58 pm

nửa trên với nửa dưới của ma trận là 2 hướng luôn đúng k b???

Reply

nguyenvanquan7826 says:
April 20, 2015 at 6:04 am

Dúng rồi bạn, bạn chỉ cần cho 2 nửa khác nhau là thành đồ thị có hướng mà. VD 1->2 nhưng 2 ko đi đến 1 thì A[1][2] = 2, còn A[2][1] = vô cùng.

Reply

duyệt says:
April 20, 2015 at 2:04 pm

a có thể làm mô phỏng về thuật toán này không?? giải thích rõ code bằng mô phỏng a?? e mới học lập trình mong a giúp.

Reply

nguyenvanquan7826 says:
April 20, 2015 at 9:24 pm

Bạn có thể xem ở đây nhé
http://nguyenvanquan7826.com/2014/06/14/java-thuat-toan-mo-phong-thuat-toan-dijkstra-tim-duong-di-ngan-nhat/

Reply

Veoo says:
April 28, 2015 at 4:51 pm

Tuyệt ! Rất thích cách trình bày của bạn.

Reply

tun says:
April 29, 2015 at 5:46 pm

bạn ơi cho mình hỏi file input.inp thì tạo thế nào và để ở đâu vậy. mình dùng visual studio 2010 chạy có đc không

Reply

nguyenvanquan7826 says:
April 29, 2015 at 6:02 pm

finle input.inp là 1 file text thôi, để cùng thư mục với file .c hoặc .cpp
VS thì mình không rõ. Mình chưa dùng nó, bạn cứ thử xem

Reply

tun says:
May 1, 2015 at 10:30 am

ok. mình cảm ơn b nhé :v

Reply

cuwowng says:
May 8, 2015 at 8:25 pm

cho mình hỏi thuật toán warshall này của mình viết kiêu này nó chỉ hiển thị đúng từ w0 đến p6 là kết thúc mà không thấy hiển thị đường đi ngắn nhất từ đâu đến đâu cả . bạn xem và có thể sửa hộ giúp mình được k.
code :

```
#include
#include
int main()
{

int w[10][10];
int k, n, i, j;
int p[10][10];
printf("nhap vao kich thuong n:"); scanf("%d",&n);
//nhap vao w[0] va p[0]
for (i=0; i<n; i++) for (j=0; jw[0]:n");
for (i=0; i<n; i++)
{
for (j=0; jp[0]:n");
for (i=0; i<n; i++)
{
for (j=0; j<n; j++) if (p[i][j]!=32) printf("%3c",p[i][j]+64); else printf("%3c",32);
printf("n");
}
getch();
//tinh toan ra in ra w[k],p[k] voi k=1,2,...,k
for (k=0; kn w[%d]: p[%d]:n",k+1,k+1);
for (i=0; i<n; i++)
{
for(j=0; jw[i][k]+w[k][j])
{
w[i][j]=w[i][k]+w[k][j];
p[i][j]=p[i][k];
```

```
}
printf (“%3d”,w[i][j]);
}
printf (” || “);
for (j=0; j<n; j++)
if(p[i][j]==32) printf("%3c",p[i][j]);
else printf("%3c",p[i][j]+64);
printf("||"); printf("n");
}
}
}
//*tim ra duong di ngan nhat
printf("Nhap dinh xuat phat s = "); scanf("%d",&s);
printf("Nhap dinh ket thuc f = "); scanf("%d",&f);
Dijkstra(n,a,L,pi,s);
if(L[f]==vc) printf("Khong co duong di");
else
{
printf("n Duong di tu %d den %d ngan nhat %d n",s,f,L[f]);
induong(s,f,pi);
}
delete L;
delete pi;
getch();

}
```

Reply

nguyenvanquan7826 says:
May 8, 2015 at 10:23 pm

Cái thuật này mình chưa tìm hiểu nên xin lỗi bạn mình chưa rõ 🙂

Reply

ducmanhkthd says:
May 9, 2015 at 9:19 pm

anh cho em hỏi,muốn đánh giá thuật toán Floyd như thế nào ạ? em chỉ biết nó ra 0(n^3) anh có thể hướng dẫn em đánh giá thuật toán này ko?thank ạ

Reply

nguyenvanquan7826 says:
May 9, 2015 at 9:33 pm

Bạn xem cách đánh giá ở bài này nhé:
http://www.nguyenvanquan7826.com/2013/06/14/thuat-toan-p1-cach-tinh-do-phuc-tap-thuat-toan-algorithm-complexity/

Reply

cương says:
May 10, 2015 at 9:15 am

cho mình hỏi cái code nâng cao ở trên sao mình chạy trên code block bị lỗi gì ý các chữ chạy loạn lên

Reply

nguyenvanquan7826 says:
May 10, 2015 at 10:49 am

Mình chạy vẫn ổn mà, nó báo lỗi thế nào bạn?

Reply

cương says:
May 10, 2015 at 1:46 pm

Reply

nguyenvanquan7826 says:
May 11, 2015 at 12:07 am

Do bạn đặt file là *.c nên nó không phải c++ nên không có các thư viện đó. Các thư viện đó là của C++ bạn phải đặt file là *.cpp

Reply

duyệt says:
May 19, 2015 at 6:16 pm

cho e hỏi tại sao floyd dùng được trọng số âm còn dijkstra lại không?? nói rõ e được không?? e k hiểu

Reply

nguyenvanquan7826 says:
May 20, 2015 at 9:31 am

Chào bạn, xin lỗi bạn hôm qua khi bạn gọi cho mình và mình đã khẳng định sai lệch khi cho rằng thuật toán vẫn đúng với trọng số âm. Để biết về cái này chúng ta cần dựa vào cách chứng mình thuật toán:
"Chúng ta sẽ chỉ ra, khi một đỉnh v được bổ sung vào tập S, thì d[v] là giá trị của đường đi ngắn nhất từ nguồn s đến v.

Theo định nghĩa nhãn d, d[v] là giá trị của đường đi ngắn nhất trong các đường đi từ nguồn s, qua các đỉnh trong S, rồi theo một cạnh nối trực tiếp u-v đến v.

Giả sử tồn tại một đường đi từ s đến v có giá trị bé hơn d[v]. Như vậy trong đường đi, tồn tại đỉnh giữa s và v không thuộc S. Chọn w là đỉnh đầu tiên như vậy.

Đường đi của ta có dạng s – … – w – … – v. Nhưng do trọng số các cạnh không âm nên đoạn s – … – w có độ dài không lớn hơn hơn toàn bộ đường đi, và do đó có giá trị bé hơn d[v]. Mặt khác, do cách chọn w của ta, nên độ dài của đoạn s – … – w chính là d[w]. Như

vậy d[w] < d[v], trái với cách chọn đỉnh v. Đây là điều mâu thuẫn. Vậy điều giả sử của ta là sai. Ta có điều phải chứng minh." Như vậy nếu trọng số của các cạnh có thể âm thì ta không khẳng định được "đoạn s - ... - w có độ dài bé hơn đoạn s-w-v" và như vậy ta không thể chứng minh tiếp. Trong khi đó xét thuật toán floyd: "Để đi từ a --> b. Bạn mất 1 quãng đường là x. Thuật toán sẽ tìm 1 đường đi gián tiếp từ a — k — b và nếu đường đi này ngắn hơn đường đi trực tiếp thì ta gán luôn giá trị nhỏ nhất của đường đi trực tiếp bằng đường đi gián tiếp." Do việc tìm đường đi gián tiếp ngay từng bước mà không tìm ngắn nhất từng đoạn như Dijkstra nên sẽ không bị ảnh hưởng bởi trọng số âm hay dương.

Reply

Tiến says:
May 26, 2015 at 7:08 pm

Bạn ơi cho mình hỏi là tìm đường đi ngắn nhất khác với tìm đường đi có trọng số nhỏ nhất à. Có cần quan tâm đến trọng số k nhỉ?

Reply

nguyenvanquan7826 says:
May 26, 2015 at 11:49 pm

Về bản chất thì khác nhưng các bài toán chúng ta xét thì hầu như coi trọng số chính là quãng đường đi nên không khác.

Reply

Tiến says:
May 27, 2015 at 1:07 am

Cho mình hỏi có cần tìm V+ – không?

Reply

nguyenvanquan7826 says:
May 27, 2015 at 1:09 am

V+- là gì bạn?

Reply

thường says:
November 1, 2015 at 10:22 pm

bạn ơi bạn có code cua thuật tóan FORD & FULKERSON ko cho minh xin

Reply

nguyenvanquan7826 says:
November 2, 2015 at 12:36 am

Minh khong co ban ah.

Reply

tỉnh ngố says:
May 10, 2016 at 11:42 pm

bạn có code c cho bài trên cho mình xin vf mình chưa học c++

Reply

nguyenvanquan7826 says:
May 11, 2016 at 12:53 am

Bài kia mình code c mà.

Reply

hoang says:
May 14, 2016 at 10:36 pm

sao e chạy thử code dijkstra phần 1 lại bị lỗi stop working nhỉ?? a xem lại đc ko ạ?

Reply

nguyenvanquan7826 says:
May 19, 2016 at 10:53 pm

Bạn check xem lỗi gì nhé.

Reply

Nguyễn Việt Dũng says:
June 3, 2016 at 3:08 am

Bạn ơi sao của mình nó cứ báo lỗi thế này nhỉ
http://www.mediafire.com/view/aowfsuq6jpz5a0c/Loi.jpg

Reply

nguyenvanquan7826 says:
June 20, 2016 at 9:43 pm

Mình ko code bằng vs c nên ko rõ.

Reply

noname says:
August 12, 2016 at 7:24 pm

Hình như code Java này bạn sưu tầm của tác giả nào đó!

Reply

nguyenvanquan7826 says:
August 17, 2016 at 11:28 am

Mình đâu có coe java trong bài này bạn? 😛 Và đảm bảo tất cả các code trong bài này và code Demo chương trình đồ họa bằng Java đều do tay mình viết. Nếu bạn thấy ở chỗ khác thì chỉ có họ copy của mình thôi. 😀 😀

Reply

ThongDH says:
October 15, 2016 at 12:36 am

Chào Quân

Chỗ này code có vấn đề :

*G = (int **) malloc((*n) * sizeof(int));

Sửa: sizeof *int

*G = (int **) malloc((*n) * sizeof(*int));

Có thể bạn run trên Os 32 bit nên kết quả vẫn đúng, trên 64bit chắc là có vấn đề.
Mình chỉ đọc qua thôi chứ chưa debug :))

Thân.

Reply

nguyenvanquan7826 says:
October 15, 2016 at 1:01 am

Ah rồi, mình có thấy lỗi. 🙂 cảm ơn bạn nhé.

Reply

le tran phuong nhi says:
October 16, 2016 at 12:27 pm

a có thể mô phỏng thuật toán AT giúp e dc k?

Reply

nguyenvanquan7826 says:
October 16, 2016 at 9:14 pm

AT la thuat toan gi the?

Reply

Tiến Mướp says:
November 4, 2016 at 2:30 pm

MÌnh vừa review code thì thấy 1 điểm khá là thắc mắc là :
Đoạn code này bạn gán:
int n, a, b, i, sum = 0; // ==> sum = 0 ;
for (int i = 0; i Len[0] = 0 , Len[1] = 0 …. Len[n-1] =0;
}
Len[a] = 0;
// đoạn ở giữa này không có chỗ nào gán lại Len[i] mà i chạy từ 0 đến n-1 ;
//==> Len[0] = 0 , Len[1] = 0 …. Len[n-1] =0;
for (int j = 0; j < n; j++) { // tinh lai do dai cua cac diem chua xet
if (!S[j] && Len[i] + G[i][j] < Len[j]) {
Len[j] = Len[i] + G[i][j]; // thay doi len
??? cái điều kiện Len[i] + G[i][j] < Len[j] có bao giờ sảy ra không? khi mà 0 + k < 0 ?

Reply

nguyenvanquan7826 says:
November 18, 2016 at 5:19 pm

Bạn xem lại code nhé, mình gán cho Len ban đầu là sum trong vòng for
Len[i] = sum;

Reply

thu says:
November 14, 2016 at 7:53 am

a thật nhiệt tình.. cảm ơn vì bài viết

Reply

Do chi says:
November 17, 2016 at 9:47 am

Mk dùng code block nhưng sao k chay đc hàm con du mk đã khai báo đầu đủ rồi .

Reply

nguyenvanquan7826 says:
November 27, 2016 at 6:37 pm

Bạn nhìn xem nó báo lỗi gì nhé, dựa vào đó mới sửa được 🙂

Reply

Huy Thông says:
November 18, 2016 at 3:51 am

Hay

Reply

Phương says:
May 1, 2017 at 9:47 am

sao mình chạy code nó báo là không tìm thấy file input là sao vậy ad, chỉ e với

Reply

nguyenvanquan7826 says:
May 5, 2017 at 12:02 am

Bạn nhớ để file input.inp cùng thư mục với file code nhé.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

POST COMMENT

CLB TIN HỌC ICTU
tinhoc.ictu@gmail.com 0963 101 548