ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

BÁO CÁO THỰC HÀNH
CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Challenge 2: HAMILTONIAN GRAPH AND THE TRAVELING SALESMAN PROBLEM

Chung Hoàng Tuấn Kiệt	19120553
Nguyễn Đức Hạnh	19120011
Lê Trung Hiếu	19120507
Trịnh Nguyên Hưng	19120015

Giảng viên: Bùi Huy Thông

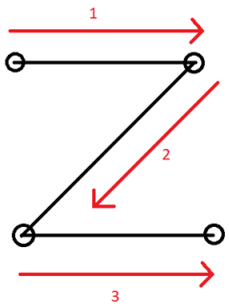
Thành phố Hồ Chí Minh, tháng 1 năm 2021

Mục lục

1 Định nghĩa

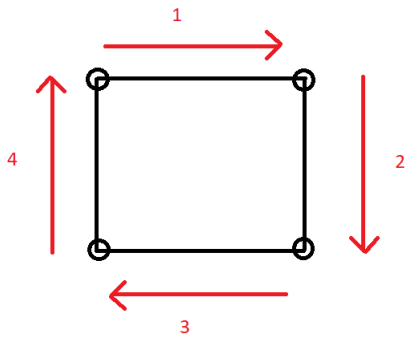
1.1 Hamilton Path

Đường đi Hamilton (Hamilton Path) là đường đi qua toàn bộ các đỉnh của đồ thị, và mỗi đỉnh chỉ đi qua đúng 1 lần.



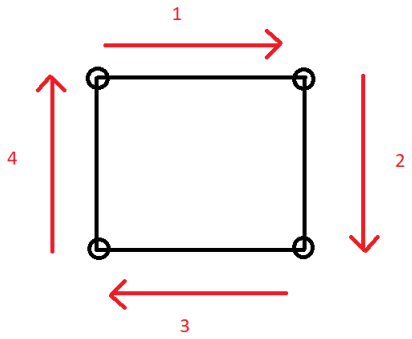
1.2 Hamilton Cycle

Chu trình Hamilton (Hamilton Cycle) là chu trình bắt đầu từ 1 đỉnh nào đó của đồ thị, đi qua tất cả các đỉnh còn lại của đồ thị, mỗi đỉnh chỉ đi qua đúng 1 lần rồi lại trở về điểm xuất phát.



1.3 Hamilton Graph

Đồ thị Hamilton (Hamilton Graph) là đồ thị có chứa chu trình Hamilton.



Nếu 1 đồ thị chỉ chứa đường đi Hamilton mà không chứa chu trình Hamilton thì nó được gọi là đồ thị nửa Hamilton (hình 1 là đồ thị nửa Hamilton vì nó chứa đường đi Hamilton mà không chứa chu trình Hamilton).

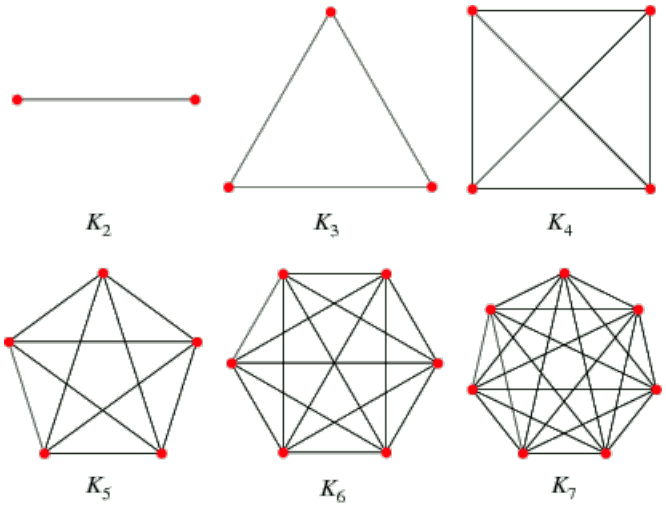
2 Maximum and minimum number of Hamiltonian Cycle(s)

Một đồ thị có kích thước n bất kì, số chu trình Hamilton tối thiểu của nó rõ ràng là 0 với trường hợp không tìm được chu trình Hamilton nào trong đồ thị đó.

Số chu trình Hamilton tối đa của một đồ thị là n! với trường hợp đồ thị đó là đồ thị đầy đủ (complete graph), tức là đồ thị mà giữa 2 đỉnh bất kì luôn có cạnh nối.

Với một đồ thị đầy đủ có n đỉnh ,ta có n cách chọn đỉnh thứ nhất. Sau đó chọn một đỉnh bất kỳ mà không phải đỉnh thứ nhất ta có n - 1 cách (vì từ 1 đỉnh bất kỳ của đồ thị đầy đủ luôn có đường đi tới các đỉnh còn lại), rồi lại chọn tiếp một đỉnh trong số đỉnh còn lại có n - 2 cách, cứ như thế đến cuối cùng ta có n*(n -1)*(n - 2)*. . . *1=n! cách để tạo nên một chu trình Hamilton từ n đỉnh của đồ thị đầy đủ.

Một số ví dụ về đồ thị đầy đủ:



3 Complexity to verify a graph is hamiltonian

Để xác định một đồ thị là Hamiltonian hay không ta sử dụng giải thuật backtracking(quay lui).

Một số quy ước tên biến và chức năng của biến trong hàm Check-Hamiltonian.

- **Biến x:** là một mảng các số nguyên để kiểm tra và lưu lại kết quả đỉnh hiện tại có đường đi đến các đỉnh khác hay không .
- **Biến k:** Đã đi đến được đỉnh thứ k.
- **Biến n:** Số đỉnh của đồ thị.
- **Biến G:** Mảng 2 chiều để lưu lại các cạnh của đồ thị.
- **Biến Check:** Biến kiểu bool có tác dụng dừng kiểm tra khi đã xác định được tồn tại Hamilton cycle.

Các bước thực hiện:

- **Bước 1:** Ta tiến hành kiểm tra giá trị của biến check. Nếu là true thì ra thoát khỏi hàm hiện tại. Nếu giá trị biến check là false thì ta tiến hành vòng lặp ở bước 2.
- **Bước 2:** Ta thực hiện vòng lặp while (true). Từ đỉnh hiện tại, ta tìm ra một đỉnh sao cho tồn tại cạnh nối giữa đỉnh hiện tại và đỉnh đó. Kết quả tìm được ta lưu vào phần tử tại vị trí k trong mảng x.
- **Bước 3:** Nếu phần tử tại vị trí k trong mảng x có giá trị là 0. Đồng nghĩa với việc đỉnh chúng ta đang tìm là đỉnh lá hoặc các đỉnh liên kết với nó đã được duyệt qua thì ta dừng kiểm tra và thoát khỏi hàm hiện tại. Nếu x[k]!=0 , ta sang bước 4.
- **Bước 4:** Nếu giá trị hiện tại của k là n thì ta gán check = true. Nếu sai ta sang bước 5.
- **Bước 5:** Ta gọi đệ quy hàm Check-Hamilton hiện tại với giá trị k = k + 1.

Mã giả:

```
NextVertex(x, k, n, G) {
  do {
        x[k] = (x[k] + 1) % (n+1)
    if (x[k] = 0)
        return

      if (G[x[k - 1]][x[k]]) {
        for j from 1 to k - 1
          if x[j] = x[k]
              break
        if j = k
          if k < n or (k = n and G[x[n]][x[1]])
              return
      }
  } while (true)
}

Hamilton_Cycle(x, k, n, G, check) {
  if (check)
      return
  do {
      NextVertex(x,k,n,G)
    if (x[k] = 0)
        return
    if (k = n)
      check = true
    else
        Hamilton_Cycle(x, k + 1, n,G, check)
  } while (true)
}
```

Chúng ta sẽ bắt đầu chu trình hamilton ở đỉnh 1. Tai mỗi đỉnh ta tiến hành kiểm tra sự tồn tại đường đi từ điểm hiện tại đến đỉnh đó và đỉnh đó đã được thăm hay chưa thông qua hàm **Next-Vertex**. Sau khi ta tìm được điểm thích hợp ta tiếp tục gọi đệ quy đến khi duyệt qua hết các đường đi thì ta quay lui về đỉnh hiện tại và sang đỉnh khác.

Từ đỉnh ban đầu chúng ta sẽ duyệt qua n - 1 đỉnh còn lại. Ở lần duyệt thứ hai, chúng ta sẽ tiếp tục xét n - 2 đỉnh còn lai. Chúng ta sẽ duyệt như thế đến khi xác đỉnh tồn tại một chu trình hamilton thì dừng lại. Nếu không tồn tại chu trình nào thì trả về false và thoát khỏi chương trình. Vậy độ phức tạp của thuật toán chúng ta là:

$$(n-1)*(n-2)*...*1 = (n-1)! \tag{1}$$

Trong trường hợp tốt nhất, đồ thị mà chúng ta đang xét là một đồ thị đầy đủ thì số bước thuật toán của chúng ta phải thực hiện để tìm ra một chu trình hamilton là:

$$n*(n-1) \tag{2}$$

Chứng minh:
Do đồ thị đang xét là một đồ thị đầy đủ và thuật toán của chúng ta xét theo thứ tự đỉnh nên từ đỉnh 1 ta sẽ gọi đệ quy sang đỉnh 2. Tại đỉnh 2, ta tiếp tục lời gọi đệ quy đến đỉnh 3.

Ta tiếp tục thực hiện đến đỉnh n thì dừng. Do ta đã tìm ra chu trình hamilton nên biến check sẽ mang giá trị là **true**. Sau đó chúng ta quay lui về đỉnh n - 1. Ta gọi đệ quy và xét thấy biến check đã mang giá trị là **true** nên chúng ta quay lui trở lại. Ở lời gọi đệ quy thứ n - 1 sẽ không thực hiện thêm bất kì thao tác nào.

Trong lời gọi đệ quy thứ n - 2 sẽ dừng khi duyệt qua toàn bộ n đỉnh. Tiếp tục quay lui lên các lời gọi đệ quy lần thứ n - 3. Ta xét thấy biến check đã mang giá trị true. Nên thuật toán sẽ chỉ gọi đệ quy và thoát hàm đệ quy đó ngay khi kiểm tra giá trị biến **check**. Tương tự xét đến các lời gọi đệ quy quay lui trước.

Vậy sau khi đã tìm được chu trình hamilton, các lời gọi đệ quy của chúng ta sẽ chỉ duyệt qua các đỉnh mà không thực hiện thêm bất kì thao tác nào. Do đó, ở mỗi lượt duyệt chúng ta sẽ tốn chi phí là n - 1. Sau n - 1 lần thực hiện thì chi phí thuật toán là

$$(n-1)^2 \tag{3}$$

4 Travelling Salesman Problem - TSP (Bài toán người bán hàng)

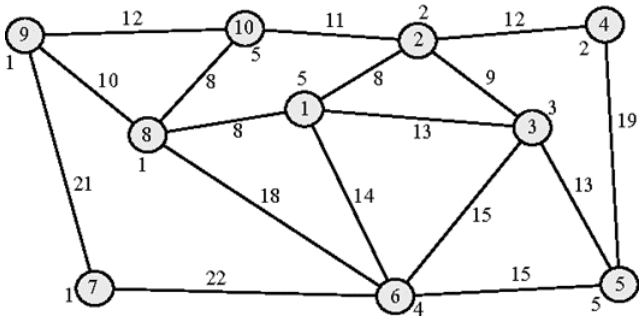
Người bán hàng muốn đi giao hàng qua n thành phố. Anh ta xuất phát từ một thành phố nào đó và đi qua các thành phố khác để giao hàng sau đó trở về thành phố ban đầu. Mỗi thành phố chỉ đến một lần duy nhất và khoảng cách từ một thành phố đến các thành phố khác đã được biết trước. Vấn đề của bài toán là chúng ta phải tìm một đường đi thỏa mãn các điều kiện trên sao cho tổng độ dài quãng đường là nhỏ nhất.

TSP có một vài ứng dụng trong dạng thức nguyên thuỷ của nó như lập kế hoạch, logistic, và sản xuất các microchip. Thay đổi đi chút ít nó xuất hiện như một bài toán con trong rất nhiều lĩnh vực như việc phân tích gen trong sinh học. Trong những ứng dụng này, khái niệm thành phố có thể thay đổi thành khách hàng, các điểm hàn trên bảng mạch, các mảnh DNA trong gen, và khái niệm khoảng cách có thể biểu diễn bởi thời gian du lịch hay giá thành, hay giống như sự so sánh giữa các mảnh DNA với nhau. Trong nhiều ứng dụng, các hạn chế truyền thống như giới hạn tài nguyên hay giới hạn thời gian thậm chí còn làm cho bài toán trở nên khó hơn.

Mối liên hệ giữa bài toán người bán hàng và chu trình Hamilton:.

Tuyến đường người bán hàng cần đi chính là một chu trình Hamilton, bắt đầu từ một thành phố, đi qua tất cả các thành phố còn lại, mỗi thành phố chỉ đi qua đúng 1 lần rồi trở về thành phố ban đầu, nó giống với định nghĩa của chu trình Hamilton là chu trình bắt đầu từ 1 điểm (ứng với 1 thành phố), đi qua các đỉnh còn lại (các thành phố còn lại) của đồ thị (các tuyến đường) rồi trở về điểm ban đầu (thành phố ban đầu). Điểm khác nhau là nếu tồn tại nhiều chu trình hamilton với chi phí của chu trình (ứng với chi phí đi qua toàn bộ các thành phố và trở về thành phố ban đầu) khác nhau thì chúng ta phải chọn chu trình có chi phí thấp nhất.

Để biểu diễn bài toán dưới dạng đồ thị, danh sách các thành phố và các con đường nối các thành phố là một đồ thị vô hướng có trọng số (undirected weighted graph), tức là với mỗi cạnh được gán với một giá trị, một số nào đó biểu diễn khoảng cách giữa 2 thành phố. Bài toán sẽ tương ứng với việc tìm chu trình Hamilton trong đồ thị vô hướng có trọng số sao cho tổng độ dài các cạnh trong chu trình đó là ngắn nhất..



5 Complexity to find Traveling Salesman Tour from a connected undirected graph

Tuyến đường người bán hàng cần đi chính là một chu trình Hamilton (câu trên), do đó có thể áp dụng thuật toán để tìm chu trình Hamilton. Tuy nhiên vì cần tìm đường đi với chi phí nhỏ nhất nên chúng ta sẽ không dừng lại khi tìm được 1 chu trình hamilton mà cần phải xem xét tất cả các chu trình hamilton có thể có. Chúng ta có tối đa n! chu trình Hamilton nên trong trường hợp xấu nhất, độ phức tạp là $O(n!*2^n)(4)$ (xét theo thuật toán đã trình bày bên trên) nếu tính toán gọn mà không phát sinh thêm chi phí phức tạp xử lý sau khi tìm được 1 chu trình nào đó. Nên tính chi phí quãng đường đồng thời với việc thêm cạnh khi tìm chu trình để tiết kiệm thời gian.

Ngoài giải thuật đơn giản trên còn có 1 giải pháp khác đưa ra kết quả chính xác là quy hoạch động. Đây là cách giải quyết bài toán dựa trên trạng thái, hay còn gọi giải pháp này là quy hoạch động trạng thái. Gọi DP[S][i] là chi phí tối thiểu để thăm các đỉnh với trạng thái S, tức là bit thứ k trong S mà bằng 1 có nghĩa là đỉnh k đã được thăm. Và i là đỉnh cuối cùng được thăm trong trạng thái S. Ta có D[S][i] = min(D[P][j] + C[j][i]) với P là S với bit i = 0, j là đỉnh cuối khi thăm với trạng thái P, ta thay j là các đỉnh đã thăm trong trạng thái P. Với cách làm này, ta dễ dàng tìm ra được kết quả cuối cùng. Độ phức tạp của thuật toán liên quan tới số lượng trạng thái, có tất cả là $2^n(5)$ trạng thái, cùng với duyệt với 2 biến i, j nên độ phức tạp là

$$0(2^n * n^2) \tag{6}$$

Một số quy ước tên biến và chức năng của biến trong hàm TSP.

- **Biến start:** Đỉnh bắt đầu của chun trình.
- **Biến mask:** Là một biến check giúp chúng ta đánh dấu những đỉnh đã duyệt qua .

- **Biến pos:** Đỉnh tại vị trí đang xét.
- **Biến T:** Mảng 2 chiều để lưu sự liên kết giữa các đỉnh trong đồ thị.
- **Biến G:** Mảng 2 chiều để lưu lại trọng số các cạnh của đồ thị.
- **Biến VISITED_ALL:** Giá trị để kiểm tra khi đã duyệt qua tất cả các đỉnh của đồ thị.
- **Biến DP:** Mảng 2 chiều lưu lại đường đi của thuật toán TSP.

```
TSP(start,mask, pos, n, T, G, VISITED_ALL,DP){
 if mask=VISITED_ALL {
  if (T[pos][start]) {
   DP[mask][pos] = G[pos][start]
   return G[pos][start]
  }
  else {
   DP[mask][pos] = INF
   return INF
  }
 }
 if DP[mask][pos] != -1 {
  return DP[mask][pos]
 }
 ans = INF
 for i from 0 to n {
  if T[pos][i] && (mask & (1 << i)) = 0 {
   MID = TSP(start, mask | (1 << i), i, n, T,G, VISITED_ALL, DP) + G[pos][i]
   if ans > MID {
    ans = MID
   }
  }
 }
 DP[mask][pos] = ans
 return ans
}
```

6 Solution of the TSP problem

Trong bài viết, thuật toán này chỉ đúng khi các cạnh của đồ thị thỏa mãn bất đẳng thức trong tam giác thì giá trị của nó không đến bằng 2 lần giá trị của đường đi đúng. Với 1 đồ thị bất kì không thỏa mãn đồ thị trên, nhưng vẫn liên thông và vô hướng thì chúng ta sẽ quan tâm là thuật toán này có đưa ra 1 chu trình hamilton hay không vì ở đồ thị này không thể có giá trị kết quả như ở đồ thị thỏa mãn bất đẳng thức tam giác. Giả sử 1 đồ thị có 7 đỉnh, khi thực hiện thuật toán Prim để tìm cây khung nhỏ nhất, để không làm mất tính tổng quát, giả sử các đỉnh đã lần lượt được chọn để thêm vào cây khung tại thời điểm chọn, với thứ tự sau: 1 – 2 – 3 – 6 – 7 – 5 – 4. Ở đây có thể đỉnh 6 và đỉnh 7 đều được thêm vào cây khung vì cả 2 đỉnh này có cạnh nối với đỉnh số 3 là 2 cạnh nhỏ nhất trong các cạnh mà chúng ta xét tại thời điểm đó. Tuy nhiên trong trường hợp này, chúng ta chỉ biết 6 và 7 đều có cạnh trực tiếp với 3 mà không biết giữa cạnh 6 và 7 có đường nối trực tiếp hay không. Do đó thuật toán này chắc chắn áp dụng được trên đồ thị đầy đủ, nhưng đối với đồ thị không đầy đủ thì không.

Tài liệu

1. Traveling Salesman Problem - Wikipedia English
   https://en.wikipedia.org/wiki/Travelling_salesman_problem
2. Hamiltonian Path - Wikipedia English
   https://en.wikipedia.org/wiki/Hamiltonian_path
3. Traveling Salesman Problem
   https://cs.stackexchange.com/questions/90149/analysis-of-time-complexity-of-travelling-salesman-problem