**Thiết kế và đánh giá thuật toán :**

3. Tự đặt một số đề bài toán , phân tích bài toán , xây dựng giải thuật , đánh giá độ phức tạp và viết chương trình để minh họa phương pháp quay lui ,phương pháp nhánh cận (mỗi phương pháp ít nhất một bài toán)

**1. Quay lui :**

Bài toán tìm chu trình hamilton :

**Đề bài :**

Đồ thị vô hướng là đồ thị mà khi đỉnh A kề với B thì đỉnh B cũng kề với đỉnh A.

Chu trình hamilton : Là chu trình đi qua tất cả các đỉnh , mỗi đỉnh đúng một lần , sau đó lại quay lai đỉnh xuất phát .

Cho đồ thị G vô hướng n đỉnh , tìm chu trình hamilton của đồ thị.

**Bài toán :**

Input :

Dòng 1 ghi số đỉnh n (n < = 100) của đồ th . N dòng tiếp theo là một ma trận U[n][n] trong đó:

U[i][j] = 1 nếu I và j là 2 đỉnh kề nhau

U[i][j] = 0 nếu hai đỉnh i và j không kề nhau.

Out put :

File văn bản liệt kê các chu trình hamilton.

**Ý tưởng :**

Ta dùng phương pháp quay lui . Đây là quá trình tìm theo chiều sâu với một tập hợp các lời giải . Trong quá trình tìm kiếm lời giải nếu ta gặp một hướng không thỏa mãn , thì sẽ không đi theo hướng đó nữa , ta quay lui về hướng có các hướng lựa chọn tiếp theo. Khi đã thực hiện hết các khả năng từ điểm đó ta sẽ quay về điểm trước đó và thử các hướng chọn tiếp theo từ điểm đó .Quá trình tìm kiếm dừng lại khi không còn điểm nào được lựa chọn .

**Xây dựng thuật giải:**

Ban đầu ta xuất phát từ một đỉnh (ta chọn đỉnh 1)

Sau đó duyệt tất cả các đỉnh còn lại (theo thứ tự 1 - > n) nếu bắt gặp đỉnh kề với đỉnh đó và chưa có trong chu trình thì ta thêm vào chu trình . Rồi gọi hàm hamilton với điểm vừa tìm được . Nếu chu trình đủ n điểm và quay về điểm xuất phát thì ta in chu trình đó ra và thực hiện tiếp cho tới khi không còn điểm nào được chọ nữa ta sẽ dừng vòng lặp lại.

**Code :** Có trong file Hamilton.cpp

**2. Nhánh cận .**

**Đề bài :** Cho trước một danh sách các thành phố và khoảng cách giữa chúng , tìm chu trình ngắn nhất đi được tất cả các thành phố, mỗi thành phố đi qua một lần rồi quay trở lại thành phố xuất phát sao cho khoảng cách đường đi là ngắn nhất.

**Bài toán:**

In put :

Dòng 1 : Cho n là số đỉnh.

Tiếp theo là ma trận n dòng là khoảng cách của từng điểm đến các điểm còn lại.

Output : Phương án tối ưu cho người bán hàng và khoảng cách ngắn nhất.

**Ý tưởng :**

Giữ lại một phương án mẫu .

Tính các phương án khác ngay trong quá trình xây dựng .

Nếu có phương án tốt hơn ta cập nhật lại và đi đến phương án tiếp theo .

Nếu không tốt hơn quay lại phương án trên xét phướng án tiếp theo.

Cho đến khi nào không có phương án nào không còn phương án nào nữa ta sẽ thu được phương pháp tốt nhất.

**Xây dựng thuật giải:**

Ban đầu ta sẽ có một ma trận vuông n\* n .a[n][n].

Khởi tạo mảng đánh dấu dd[max], mảng phương án tạm thởi pa[max], một mảng phương án tối ưu paToiUu[max].

Một biến khoảng cách tạm thời kc, và khoảng cách tối ưu (ngắn nhất) kcMin

Vì là đồ thị vô hướng và có chu trình nên ta chọn đỉnh bất kỳ để xuất phát (ta chọn đỉnh 1).

Sau đó lần lượt duyệt các đỉnh tiếp theo nếu đỉnh đó kề với đỉnh đang xét và chưa được đánh dấu ta sẽ cho đỉnh đó vào chu trình và đánh dấu đỉnh đó lại , cập nhật khoảng cách .

Nếu ta đã xét hết các đỉnh và đỉnh cuối kề với đỉnh đầu thì ta lưu đó là một phương án. So sánh với phương án trước đó nếu nó tốt hơn thì ta cập nhật phương án tối ưu và khoảng cách ngắn mới.

Sau khi thực hiện tất cả các phương án thì paToiUu[] là mảng lưu lại chu trình tối ưu và kcMin là khoảng cách tối ưu.

**Code :** Có trong file NguoiBanHang.cpp

Bài 2.4. Bài toán xâu ABC : Cho số nguyên dương N < 100 , tìm xâu gồm các ký tự A, B, C thỏa mãn : Xâu có độ dài N , hai đoạn con bất kỳ gần nhau đều cách nhau, xâu có ít ký tự C nhất .

Bài toán :

Bài toán tìm tất cả các từ đẹp độ dài n cho trước yêu cầu tìm nghiệm là các

vector x có n thành phần:

1. xi nhận giá trị trong tập {A,B,C}

2. x không có 2 đoạn con liên tiếp nào bằng nhau.

Phân tích bài toán :

Nếu dãy X[1…..n] thỏa mãn điều kiện 2 dãy con bất kỳ khác nhau thì 4 ký tự liên tiếp bao giờ cũng sẽ chứa một ký tự C .Như vậy với một dãy gồm k ký tự bao giờ cũng có >= k div 4 ký tự C.Tại bước thử chọn X[i] nếu ta đã có T[i] ký tự C thì n – X[i] còn lại có ít nhất (n-X[i]) div 4 ký tự C .

Như vậy số ký tự C trong dãy luôn luôn >= T[i] + (n-X[i]) div 4 ký tự C.

**Code :** Có trong file nguồn XauABC\_NhanhCan.cpp