CBUS

- There are n passengers 1, 2, …, n. The passenger i want to travel from point i to point $i + n$ (i = 1,2,…, n). There is a bus located at point 0 and has k places for transporting the passengers (it means at any time, there are at most k passengers on the bus). You are given the distance matrix c in which c(i,j) is the traveling distance from point i to point j (i, j = 0,1,…, 2n). Compute the shortest route for the bus, serving n passengers and coming back to point 0.
- **Input**
- Line 1 contains n and k ($1 \le n \le 11, 1 \le k \le 10$)
- Line i+1 (i = 1, 2,…, 2n+1) contains the $(i-1)^{th}$ line of the matrix c (rows and columns are indexed from 0, 1, 2,.., 2n).
- **Output**
- Unique line contains the length of the shortest route.

CBUS

stdin	stdout
3 2 0 8 5 1 10 5 9 9 0 5 6 6 2 8 2 2 0 3 8 7 2 5 3 4 0 3 2 7 9 6 8 7 0 9 10 3 8 10 6 5 0 2 3 4 4 5 2 2 0	25

CBUS - Hint

- Áp dụng kỹ thuật quay lui để duyệt tất cả các phương án, kết hợp Branch and Bound để loại bỏ các tính toán dư thừa
- Biểu diễn lời giải: $x_1, x_2, \ldots, x_{2n}$ là chuỗi $2n$ điểm (hoán vị của 1, 2, . . ., $2n$) đón – trả của hành khách trong hành trình xe bus.
- Biến phụ trợ
- Mảng đánh dấu *appear*[1..2*n*], trong đó *appear*[*v*] = true có nghĩa giá trị *v* (từ 1 đến 2*n*) đã xuất hiện trong hành trình bộ phận.
- load: ghi nhận số hành khách đang có mặt trên xe (được tích lùy dần trong quá trình duyệt)

- Hàm Try(k) thử giá trị cho *x*[*k*]
- Với mỗi giá trị v hợp lệ gán cho x[k], thực hiện
- Cập nhật: load = load + 1 nếu v <= n (điểm đón) và load = load – 1 nếu v > n (điểm trả)
- cập nhật mảng đánh dấu appear[v] = true
- Nếu k = 2n thì ghi nhận một phương án, so sánh với phương án tốt nhất đã có và thực hiện cập nhật kỷ lục
- Ngược lại, gọi tiếp Try(k+1)
- Áp dụng phương pháp đánh giá cận dưới (giống bài toán TSP) để tăng tốc độ duyệt, bỏ qua các nhánh tính toán thừa

Implementation

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 100

int N;// number of requests (1,2,...,N). Request i has pickup point i and drop-off point i + N
int cap;// number of places of the bus
int A[2*MAX+1][2*MAX+1];

int x[MAX];
int appear[MAX];// marking
int load;
int f;
int f_best;
int x_best[MAX];
int cmin;
```

```
void input(){
    scanf("%d%d",&N,&cap);
    cmin = 1000000;
    for(int i = 0; i <= 2*N;  i++){
        for(int j= 0; j <= 2*N; j++){
            scanf("%d",&A[i][j]);
            if(i != j && cmin > A[i][j]) cmin = A[i][j];
        }
    }

}

int check(int v, int k){
    if(appear[v] == 1) return 0;
    if(v >N){
        if(!appear[v-N]) return 0;
    }else{
        if(load + 1 > cap) return 0;
    }



    return 1;
}
```

Implementation

```
void solution(){
    if(f + A[x[2*N]][0] < f_best){
        f_best = f + A[x[2*N]][0];
        for(int i = 0; i <= 2*N; i++) x_best[i] = x[i];
        //printf("update best %d\n",f_best);
    }
}
```

Implementation

```
void TRY(int k){
    for(int v = 1; v <=2*N; v++){
        if(check(v,k)){
            x[k] = v;
            f += A[x[k-1]][x[k]];
            if(v <= N) load += 1; else load += -1;
            appear[v] = 1;
            if(k == 2*N) solution();
            else{
                if(f + (2*N+1-k)*cmin < f_best)
                    TRY(k+1);
            }
            if(v <= N) load -= 1; else load -= -1;
            appear[v] = 0;
            f -= A[x[k-1]][x[k]];
        }
    }
}
```

Implementation

```
void solve(){
    load = 0;
    f = 0;
    f_best = 1000000;
    for(int i = 1; i <= 2*N; i++) appear[i] = 0;
    x[0] = 0;// starting point
    TRY(1);
    printf("%d",f_best);
}
void print(){
    for(int i = 0; i <= 2*N; i++) printf("%d ",x_best[i]);
}
int main(){
    input();
    solve();
  return 0;
}
```