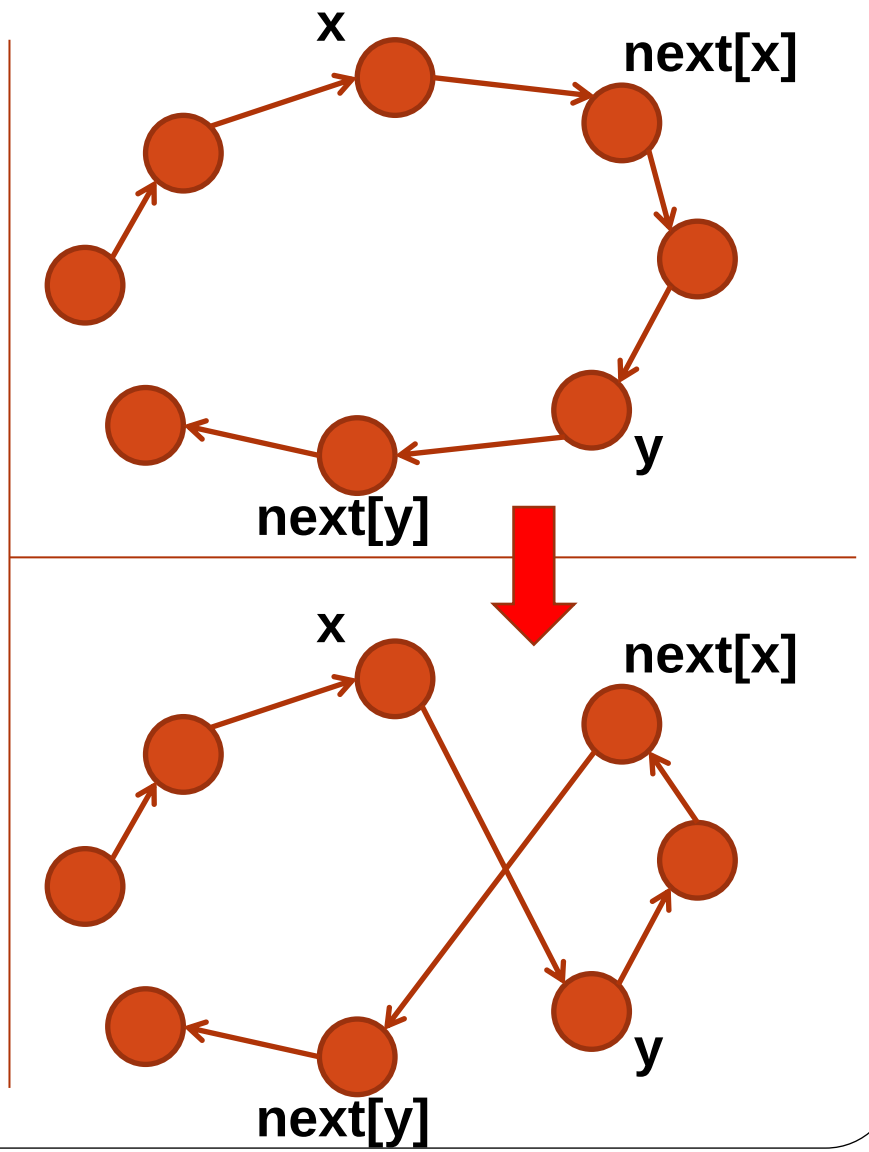
EXERCISE 9: TSP

- Solve TSP by 2-opt local search
- Use doubly linked-list for representing routes
- Design and implement
  - Initialize solution with a greedy algorithm
  - 2-opt operator
  - add and remove nodes operators

EXERCISE 9: TSP

- 2-opt operator (r, x, y)
  - r is a route
  - x, y are nodes on r, x is before y
  - r.next[x], r.next[y] are the subsequent nodes of x and y in the route r
  - r.start, r.last are starting and terminating nodes of r
  - Action
    - Remove (x, r.next[x])
    - Remove (y, r.next[y])
    - Add (x,y)
    - Add (r.next[x], r.next[y])



EXERCISE 9: TSP

```
solve(){
  r = greedyInit();
  r = twoOptImprove(r);
  return r;
}

greedyInit(){
 r = <0>;
 C = {1,2,…,N};
 while(C ≠ ∅){
   find s ∈ C such that distance(r.last, x)
                  is minimal;
   C = C\{x};
   r = r ::<x>;
 }
 r = r::<0>;
 return r;
}
```

```
twoOptImprove(r){
 while(true){
   minDis = α;  minR = ⊥;
   for(x, y ∈ r: y after x and
     r.next[y] ≠ r.last and r.next[x] ≠ y ){
      ri = twoOpt(r, x, y);
      if distance(ri)  < minDis then{
         minDis = distance(ri);
         minR = ri;
      }
   }
   if minDis < distance(r)
      r = minR;
   else break;
 }
 return r;
}
```

EXERCISE 9: TSP

- Name of the program: TSP
- Input
  - Line 1: contains integer number N
  - Line i+1: contains the ith line of the distance matrix (i = 0,…,N)
- Output:
  - Unique number which is the length of the resulting route

EXERCISE 9: TSP

- Class **Node** represents 1 point of a route, using doubly linked-list
- Class **Route** represents 1 route

```
typedef struct Node{
    int id;
    struct Node* next;
    struct Node* prev;
}TNode;

typedef struct Route{
    struct Node* start;
    struct Node* end;
    int distance;
}TRoute;
```

EXERCISE 9: TSP

`TRoute* twoOpt(TRoute* r, TNode* x, TNode* y)`	Perform 2-opt operator
`TRoute* addLast(struct Route* r, int id)`	Add a node at the end of the route
`TNode* find(TRoute* r, int id)`	Find a point having id of the given route
`TNode* makeNode(int id)`	Create a node having id
`void updateDistance(TRoute* r)`	Recompute distance of the given route
`TRoute* greedyInit()`	Initialize a route using greedy contruction
`TRoute* twoOptImprove(TRoute* r)`	Improve the solution using 2-opt operator