```
void radixSort(int *arr, int n, int max) {
   int i, j, m, p = 1, index, temp, count = 0;
   list<int> pocket[10];        //radix of decimal number is 10
   for(i = 0; i< max; i++) {
      m = pow(10, i+1);
      p = pow(10, i);
      for(j = 0; j<n; j++) {
         temp = arr[j]%m;
         index = temp/p;        //find index for pocket array
         pocket[index].push_back(arr[j]);
      }
      count = 0;
      for(j = 0; j<10; j++) {
         //delete from linked lists and store to array
         while(!pocket[j].empty()) {
            arr[count] = *(pocket[j].begin());
            pocket[j].erase(pocket[j].begin());
            count++;
         }
      }
   }
}
```

F(n): 3+6+5+2(12 N+150)

= 12 N +166

O grande:

O(n+max) es decir tiene un comportamiento lineal O(n)

Loop invariant:

J<n

```
int costominimo(int N, int M, int x[], int y[],int largox, int largoy){

    int posX=0;
    int posY=0;
    int costmin=0;

    if(largox!=(N-1) || largoy!=(M-1)){
        cout<<"Input invalido"<<endl;
        cout<<"Las listas superan los limites permitidos."<<endl;
        return 0;
    }

    for(int i=0;i<(largox+largoy);i++){

        if((x[posX]>=y[posY]) && (posX<(N-1))){ //cortar en x
            //cout<<"posX: "<<x[posX]<<endl;
            costmin=costmin + (x[posX]*(posY+1));//(posY+1) son las piezas
            posX++;
        }
        else{ //cortar en y
            //cout<<"posY: "<<y[posY]<<endl;
            costmin=costmin + (y[posY]*(posX+1));//(posX+1) son las piezas
            posY++;
        }
    }

    cout<<"\nCosto minimo de corte: "<<costmin<<endl;
    return costmin;
```

F(n): 3+6+1(4 N+ 6+5+1+5+1) + 1

= 4 N + 29

O grande:

O(largox + largoy) es decir tiene un comportamiento lineal O(n)

Loop invariant:

i<(largox+largoy)