

## Práctica 4

### Algorítmica

#### Introducción

#### Diseño del algoritmo: Branch and Bound

#### Pseudocódigo

#### Diseño del algoritmo: Backtracking

#### Pseudocódigo

#### Comparación

#### Comparación

# TSP

## Algorítmica

Universidad de Granada

1 de junio de 2016

# Índice

## Práctica 4

### Algorítmica

#### Introducción

#### Diseño del algoritmo: Branch and Bound

#### Pseudocódigo

#### Diseño del algoritmo: Backtracking

#### Pseudocódigo

#### Comparación

#### Comparación

- 1 Introducción
- 2 Diseño del algoritmo: Branch and Bound
- 3 Pseudocódigo
- 4 Diseño del algoritmo: Backtracking
- 5 Pseudocódigo
- 6 Comparación
- 7 Comparación

# Introducción

## Práctica 4

### Algorítmica

#### Introducción

Diseño del  
algoritmo:  
Branch and  
Bound

Pseudocódigo

Diseño del  
algoritmo:  
Backtracking

Pseudocódigo

Comparación

Comparación

- El objetivo de esta práctica es resolver el problema del TSP utilizando para ello un enfoque Branch and Bound y, alternativamente, otro con Backtracking y comparar ambos.

# Diseño del algoritmo

## Práctica 4

### Algorítmica

#### Introducción

#### Diseño del algoritmo: Branch and Bound

#### Pseudocódigo

#### Diseño del algoritmo: Backtracking

#### Pseudocódigo

#### Comparación

#### Comparación

Usaremos una cola con prioridad que será donde se introduzcan los nodos. En la cola se seleccionará el nodo con mayor prioridad, y de dicho nodo se consultará el valor de su cota local, que en caso de ser mejor a la global se añadirán los hijos del nodo a la cola, y en caso de ser peor, se devuelve la solución actual

# Pseudocódigo

Práctica 4

Algorítmica

Introducción

Diseño del  
algoritmo:  
Branch and  
Bound

Pseudocódigo

Diseño del  
algoritmo:  
Backtracking

Pseudocódigo

Comparación

Comparación

```
Require: Matriz_costes, Vector_ciudades[N] Vector_distancias_minimas;  
Branch&Bound( Matriz_costes, Vector_ciudades  
Vector_distancias_minimas):  
priority_queue cola;  
solucion_final;  
Nodo n.generarnodo(Vector_ciudades[0])  
while !cola.empty() do  
    nodo = cola.top();  
    if EsHoja(nodo) && (nodo.cotalocal < cota global ) then  
        solucion_final = nodo.solucion;  
        cota global = nodo.cotalocal;  
    end if  
    if (nodo.cotalocal < cota global ) then  
        cola.add(nodo.generarhijos())  
        cola.push;  
    else  
        return solucion_final;  
    end if  
end while
```

# Diseño del algoritmo

## Práctica 4

### Algorítmica

#### Introducción

#### Diseño del algoritmo: Branch and Bound

#### Pseudocódigo

#### Diseño del algoritmo: Backtracking

#### Pseudocódigo

#### Comparación

#### Comparación

La solución planteada es el mismo concepto que la aplicada en la anterior práctica, recorreremos el árbol en profundidad, calculando en cada nodo el valor de la cota local salvo que en caso de ser mayor que la global, podamos (ignorar) a ese nodo y a todos sus hijos.

# Pseudocódigo

## Práctica 4

### Algorítmica

### Introducción

### Diseño del algoritmo: Branch and Bound

### Pseudocódigo

### Diseño del algoritmo: Backtracking

### Pseudocódigo

### Comparación

### Comparación

```
Require: Matriz, S_final[N] S_parcial[N] Ciudades[N]=false ciudad_actual,  
nivel,valor_maximo=0;  
Backtrack(S,S_parcial,Ciudades,ciudad_actual,nivel):  
  Ciudades[ciudad_actual]=true;  
  S_parcial[nivel - 1]=ciudad_actual;  
  for i to N do  
    if Ciudades[i]==false then  
      valor_actual = CalcularSolucionActual(S_parcial);  
      Backtrack(S,S_parcial,Ciudades,i,nivel+1);  
      if nodo_actual == nodo_hoja then  
        valor_actual = CalcularSolucionActual(S_parcial)  
        if valor_actual mayor que valor_maximo then  
          S_final = S_Actual  
          valor_maximo = valor_actual  
        end if  
      end if  
      Ciudades[i] = false;  
    end if  
  end for
```

# Comparación

## Práctica 4

### Algorítmica

#### Introducción

#### Diseño del algoritmo: Branch and Bound

#### Pseudocódigo

#### Diseño del algoritmo: Backtracking

#### Pseudocódigo

#### Comparación

#### Comparación

En ambos programas, hemos analizado los mismos mapas.  
Compararemos los tiempos y el empleo de los nodos para cada  
tipo de técnica empleada



# Comparación de tiempos

## Práctica 4

### Algorítmica

#### Introducción

#### Diseño del algoritmo: Branch and Bound

#### Pseudocódigo

#### Diseño del algoritmo: Backtracking

#### Pseudocódigo

#### Comparación

#### Comparación

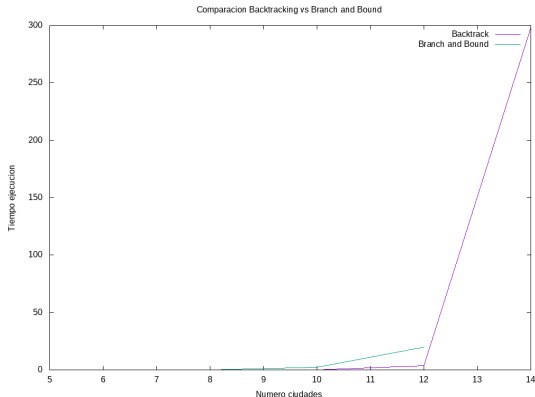


Figura: comparacion entre ambos

# Comparación de nodos: ulysses5

## Práctica 4

### Algorítmica

#### Introducción

#### Diseño del algoritmo: Branch and Bound

#### Pseudocódigo

#### Diseño del algoritmo: Backtracking

#### Pseudocódigo

#### Comparación

#### Comparación

Técnica	Branch and bound	Backtracking
N.totales	120	120
N.podados	35	68
N.explorados	85	52

# Comparación de nodos: ulysses8

## Práctica 4

### Algorítmica

#### Introducción

#### Diseño del algoritmo: Branch and Bound

#### Pseudocódigo

#### Diseño del algoritmo: Backtracking

#### Pseudocódigo

#### Comparación

#### Comparación

Técnica	Branch and bound	Backtracking
N.totales	40320	40320
N.podados	38568	37605
N.explorados	1752	2715

# Comparación de nodos: ulysses10

## Práctica 4

### Algorítmica

#### Introducción

#### Diseño del algoritmo: Branch and Bound

#### Pseudocódigo

#### Diseño del algoritmo: Backtracking

#### Pseudocódigo

#### Comparación

#### Comparación

Técnica	Branch and bound	Backtracking
N.totales	3628800	3628800
N.podados	3601453	3566077
N.explorados	27347	62723

# Comparación de nodos: ulysses12

## Práctica 4

### Algorítmica

#### Introducción

#### Diseño del algoritmo: Branch and Bound

#### Pseudocódigo

#### Diseño del algoritmo: Backtracking

#### Pseudocódigo

#### Comparación

#### Comparación

Técnica	Branch and bound	Backtracking
N.totales	479001600	479001600
N.podados	478284703	474474230
N.explorados	716897	4527370