



## Resum 2n Parcial

Anàlisis i Disseny d'Algorismes (Universitat Autònoma de Barcelona)

## Tema 4 - Branch & Bound

Algorisme:

- Selecció: Cada instant tenim llista de nodes vius
- Node: Node inicial, camí fins a ell i node destí
- Selecció: el que sembli més prometedori
- Presa de decisions: Decideix una heurística on fer cerca
- Poda: s'eliminen seqüències de decisions

B&B: Treballa amb diverses solucions parcials a les que afegeix elements per completar-les

- Nodes vius: node de l'espai de solucions del qual no s'han generat tots els seus fills
- Nodes morts: node del qual no es generaran més fills
  - No n'hi ha més
  - No és completable
  - No pot generar millor solució que la que està en curs
- Node en curs: del qual s'està generant els fills

Backtracking	Branch & Bound
-Quan es genera un fill aquest passa a ser en curs -Recorregut a cegues	-Es generen tots els fills del node en curs abans que qualsevol altre node viu passi a ser el noi node en curs -Recorregut "informat"

La llista = Cua de prioritat (quant prometedori és un node)

-Heurística: Cost + Estimació

- Observacions
  - Taula auxiliar de parels de nodes: Escriure camí de l'arrel a la solució quan afegim un node a la cua
  - Finalització de l'algorisme: si l'espai final d'estats
  - Espai d'estats infinit:
    - Si hi ha solució = garanteix finalització
    - No hi ha solució = algorisme no acaba mai
    - Es pot restringir la cerca de nodes

### Optimització i poda

Es poden tots els nodes vius de cost estimat més gran que U.

Valor inicial: heurística (informació extra sobre el problema)

Funció de cost: és mínima per tots els nodes que representen una solució òptima.

### Punts claus Branch & Bound

-Trobar bon orde de recorregut o ramificació de nodes: Funció de prioritat per trobar la solució bona ràpid

-Trobar una bona funció de poda

## **Tema 5 - Programació dinàmica**

- Divide and conquer
  - Descomposar el problema en subproblemes petits
  - Resoldre independentment
  - Combinar resultats obtinguts
  - No solapament
  - No calcular subtasques més d'una vegada
- Principi d'optimalitat de Bellman
  - Resoldre problemes d'optimització
  - Seqüència de decisions
  - Cada subseqüència ha de ser òptima
- Propietats
  - Subproblemes simples:  
Hem de poder dividir el problema original en subproblemes més petits amb la mateixa estructura
  - Subestructura òptima dels problemes (Bellman):  
La solució al problema ha de ser la composició de les solucions als subproblemes
  - Subproblemes dependents:  
L'espai de solucions dels diferents subproblemes no és disjunt
- Estratègies
  - Top-Down
    - Aplicació directa de la formulació recursiva de tots els problemes
    - Si la solució és recursiva i els problemes es solapen
      - Desem resultats de subproblemes en una taula
    - Quan volem resoldre un subproblema consultem si l'hem resultat anteriorment
- Distàncies d'edició de cadenes
  - La longitud de la subseqüència comuna més llarga:  
Només inserció i supressió
  - La distànciade Damerau-Levenshtein:  
Inserció, supressió, substitució i transposició de dos caràcters adjacents.
  - La distànciade Hamming:  
Només substitució (només s'aplica a les cadenes de caràcters de la mateixa longitud).

## Tema 6 - Probabilístics

Substituir un algorisme que dona sempre la solució correcta/òptima per un altre que de vegades l'encerta però és més ràpid.

A més recursos dedicats, millors resultats

- Més temps d'execució
- Més memòria
- Millors resultats (més probabilitat d'encertar el resultat)

Deterministes	Probabilístics	
-Sempre ha d'acabar	-Probabilitat petita es permet -Solucions diferents amb les mateixes dades	Finalització
-Sempre troba la mateixa	-Solucions diferents amb les mateixes dades	Solucions trobades
-No solucions incorrectes	-Pot equivocar-se en prob. petita -Pot repetir l'execució amb mateixes dades	Correcció Resultat
-Anàlisi difícil	-Anàlisi molt difícil	Eficiència

- Algorismes que no garanteixen la correcció de la solució
  - Algorismes Numèrics:
    - Solució aproximada
    - Major temps d'execució = millor aproximació
  - Algorismes de Monte Carlo:
    - Resposta exacta amb altra probabilitat
    - A vegades la resposta és incorrecta
    - No podem saber si la resposta és correcta
    - Es redueix la probabilitat d'error = més temps d'execució
- Algorismes que mai donen una solució incorrecte
  - Algorismes de Las Vegas:
    - Decisions a l'atzar
    - Si no trobem la solució correcta ho admetem
    - És possible intentar amb les mateixes dades fins trobar la solució correcta
  - Algorismes de Sherwood:
    - Determinista més ràpid que el pitjor dels casos
    - Uniformitzem temps d'execució en mitjana
    - Els més costosos = es resolen més ràpid
    - Els més eficients = menys eficients

Amplificació de l'avantatge estocàstica: més repeticions i augmentem la confiança de la resposta correcta

- Transformació Geomètrica
  - Punts clau en 2 imatges i es posen en correspondència:
    - Variació de colors

- Detector corner de Harris = Tensor estructural
  - Translació, rotació i escalat
- Transformació: Proximitat entre punts claus transformats
  - Original vs transformada
- Algorismes de Correspondència:
  - Determinista
    - Correspondre punts imatge A amb B
  - Las Vegas
    - Seleccionem aleatòriament 2 punts clau de A i B
    - Repetir la transformació fins trobar la correcta
    - No hi ha guany
  - Algorisme Ransac (Random Sample Consensus)
    - Repetir n vegades
      - Subconjunt aleatori de dades d'entrada
      - Trobar solució
      - Aplicar a totes les dades

## **Tema 7 - Assercions**

Fiable:

- Correcte: comportar-se d'acord amb l'especificació
- Robust: capacitat de respondre davant de casos no inclosos en l'especificació

- Especificacions per contracte
  - Precondicions: han de complir-ho les dades d'entrada de l'algorisme
  - Postcondicions: s'han de calcular després d'executar l'algorisme
- Implementació: Fa el que diu l'algorisme considerant les restriccions que pugui suposar l'ordinador
  - Assercions: Són condicions que sempre s'han de complir
    - Quan no es compleixen = Generen un missatge
  - Excepcions: es dona quan una asserció no es compleix per un mal ús de l'implementació
- Assercions
  - Precondicions
  - Postcondicions
  - Invariants de bucle: s'han de complir en cada iteració
  - Invariants de classe: ho ha de complir tot objecte de la classe
  -
- Robustesa: Donar missatge i avortar execució / Llençar una excepció
  - Detecta els trencaments de contracte de la seva especificació i reacciona controladament
  - Detecta errors o situacions inesperades i reacciona controladament