TEORIA

PROGRAMACIÓ CIENTÍFICA

T10

Cerca i Ordenació

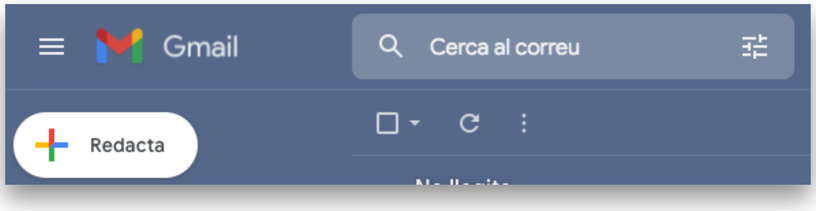
ALGORISMES DE

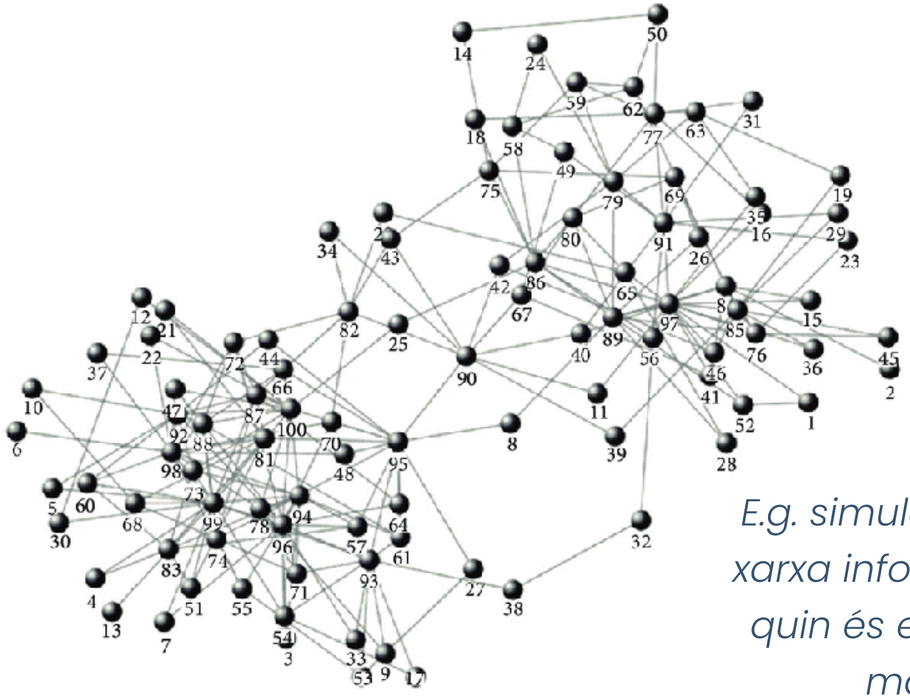
CERCA I ORDENACIÓ

Cerca

Necessitat

- Moltíssimes de les tasques automàtiques quotidianes involucren fer accions de cerca.
- Exemples: Cercar a Google, cercar dins del nostre correu, fer una consulta a una base de dades segons el nostre DNI...





E.g. simular un atac dirigit en una xarxa informàtica requereix trobar quin és el node de la xarxa amb major connectivitat.

Cerca

Algorismes de cerca

- Cerca: Indicar si un element **e** es troba dins d'un conjunt **D** (taula, fitxer, registre...etc)
- Suposarem que tenim un conjunt d'elements de tipus **registre**. I per cadascun d'ells tenim una **clau K_i** (key) que ens permet identificar-lo. El nostre objectiu és **trobar la clau** que busquem (**K_i=K**). La clau acostuma a ser el camp més identificatiu del registre.

Clau: DNI

Nom: Paquita **DNI: 12334562** Adreça: Tarragona	Nom: Jaume **DNI: 65983233** Adreça: Reus	Nom: Josefina **DNI: 52665432** Adreça: Igualada	Nom: Pepet **DNI: 39332255** Adreça: Lleida

Registres

- Nececessitem que el procés de cerca sigui **eficient**, per assegurar-nos que trobem l'element en poc temps. Ara tenim molta capacitat computacional, però per altra banda, les bases de dades són molt més grans.

Cerca

Cerca en conjunts no ordenats: Cerca seqüencial

- Primera aproximació: cerca seqüencial.
- És l'única opció si el conjunt de dades no està ordenat.

Cerca

Cerca en conjunts no ordenats: Cerca seqüencial

- Primera aproximació: cerca seqüencial.
- És l'única opció si el conjunt de dades no està ordenat.

- Consisteix en comparar la clau que busquem (**K**) amb cadascuna de les claus del conjunt de registres ($\mathbf{K_i}$) fins que el trobem (cas successful) o fins que s'arriba al final (cas unsuccessful).
- L'existència de **K** es pot assegurar tant aviat es troba però la no-existència no es pot assegurar fins haver recorregut tot el conjunt.

Cerca

Cerca en conjunts no ordenats: Cerca seqüencial

- Primera aproximació: cerca seqüencial.
- És l'única opció si el conjunt de dades no està ordenat.

- Consisteix en comparar la clau que busquem (**K**) amb cadascuna de les claus del conjunt de registres ($\mathbf{K_i}$) fins que el trobem (cas successful) o fins que s'arriba al final (cas unsuccessful).
- L'existència de **K** es pot assegurar tant aviat es troba però la no-existència no es pot assegurar fins haver recorregut tot el conjunt.

```
funció cercaSimple (vector: taula[] d'enter, mida:
enter, clau: enter) retorna booleà és
var
 i: enter;
 trobat: booleà;
fvar
inici
  i := 0;
  trobat := fals;
  mentre (i<mida i no(trobat)) fer
    si (vector[i] = clau)
      trobat := cert;
    fsi
    i:=i+1;
  fmentre
 retorna (trobat);
ffunció
```

Cerca

Cerca en conjunts no ordenats: Cerca seqüencial

- Primera aproximació: cerca seqüencial.
- És l'única opció si el conjunt de dades no està ordenat.

- Consisteix en comparar la clau que busquem (**K**) amb cadascuna de les claus del conjunt de registres ($\mathbf{K_i}$) fins que el trobem (cas successful) o fins que s'arriba al final (cas unsuccessful).
- L'existència de **K** es pot assegurar tant aviat es troba però la no-existència no es pot assegurar fins haver recorregut tot el conjunt.

- També ens pot interessar retornar la posició en la qual hem trobat l'element, per després poder accedir a la resta del registre:

```
funció cercaSimple (vector: taula[] d'enter, mida:
enter, clau: enter) retorna enter és
var
 i: enter;
 posicio: enter;
fvar
inici
  i := 0;
  posicio := -1;
  mentre (i<mida i no(trobat)) fer
    si (vector[i] = clau)
      posicio := i;
    fsi
    i:=i+1;
  fmentre
 retorna (posicio);
ffunció
```

Cerca

Cerca en conjunts no ordenats: Cerca seqüencial

Anàlisi de costos:

```
funció cercaSimple (vector: taula[] d'enter, mida:
enter, clau: enter) retorna enter és
var
 i: enter;
 posicio: enter;
fvar
inici
   i := 0;
   posicio := -1;
  mentre (i<mida i no(trobat)) fer
     si (vector[i] = clau)
        posicio := i;
     fsi
     i:=i+1;
   fmentre
 retorna (posicio);
ffunció
```

Cerca

Cerca en conjunts no ordenats: Cerca seqüencial

Anàlisi de costos:

- En el **millor cas**, l'element està a la primera posició i només fem una iteració.

  Busquem **1**:

  { 1, 5, 4 ,2 ,3 }

  A la primera iteració ja fem trobat=true

Best case = O(1)

```
funció cercaSimple (vector: taula[] d'enter, mida:
enter, clau: enter) retorna enter és
var
 i: enter;
 posicio: enter;
fvar
inici
   i := 0;
   posicio := -1;
  mentre (i<mida i no(trobat)) fer
     si (vector[i] = clau)
        posicio := i;
     fsi
     i:=i+1;
  fmentre
 retorna (posicio);
ffunció
```

Cerca

Cerca en conjunts no ordenats: Cerca seqüencial

Anàlisi de costos:

- En el **millor cas**, l'element està a la primera posició i només fem una iteració.

  Busquem **1**:

  { 1, 5, 4 ,2 ,3 }

  Best case = O(1)

  A la primera iteració ja fem trobat=true

- En el **pitjor cas**, l'element no hi és i hem de recórrer tot el conjunt.

  Busquem **6**:

  { 1, 5, 4 ,2 ,3 }

  Worst case = O(n)

  Hem de recórrer tot el vector

```
funció cercaSimple (vector: taula[] d'enter, mida:
enter, clau: enter) retorna enter és
var
 i: enter;
 posicio: enter;
fvar
inici
  i := 0;
  posicio := -1;
  mentre (i<mida i no(trobat)) fer
    si (vector[i] = clau)
      posicio := i;
    fsi
    i:=i+1;
  fmentre
 retorna (posicio);
ffunció
```

Cerca

Cerca en conjunts no ordenats: Cerca seqüencial

Anàlisi de costos:

- En el **millor cas**, l'element està a la primera posició i només fem una iteració.

  Busquem **1**:

  { 1, 5, 4 ,2 ,3 }

  Best case = O(1)

  A la primera iteració ja fem trobat=true

- En el **pitjor cas**, l'element no hi és i hem de recórrer tot el conjunt.

  Busquem **6**:

  { 1, 5, 4 ,2 ,3 }

  Worst case = O(n)

  Hem de recórrer tot el vector

- En el cas **mig**, l'element és al mig, hem de fer N/2 iteracions.

  Average case = O(n)

```
funció cercaSimple (vector: taula[] d'enter, mida:
enter, clau: enter) retorna enter és
var
 i: enter;
 posicio: enter;
fvar
inici
   i := 0;
   posicio := -1;
  mentre (i<mida i no(trobat)) fer
    si (vector[i] = clau)
      posicio := i;
    fsi
    i:=i+1;
  fmentre
 retorna (posicio);
ffunció
```

Cerca

Cerca en conjunts ordenats

Què faríeu si una persona us entrega una guia telefònica i us demana **a quina persona pertany el telèfon 977558282** ? En aquest cas no tenim més remei que fer servir la **cerca seqüencial** anterior.

En una guia telefònica, la "**clau**" és el **cognom** de la persona, i aquest conjunt de claus està **ordenat** alfabèticament.

Per aquest motiu és molt més fàcil trobar quin és el número de telèfon d'una persona que. trobar a qui pertany cert número de telèfon.

Cerca

Cerca en conjunts ordenats

Què faríeu si una persona us entrega una guia telefònica i us demana **a quina persona pertany el telèfon 977558282** ? En aquest cas no tenim més remei que fer servir la **cerca seqüencial** anterior.

En una guia telefònica, la "**clau**" és el **cognom** de la persona, i aquest conjunt de claus està **ordenat** alfabèticament.

Per aquest motiu és molt més fàcil trobar quin és el número de telèfon d'una persona que. trobar a qui pertany cert número de telèfon.

- Quan el conjunt de claus està ordenat, els mètodes de cerca poden ser més eficients.
- En el cas de la cerca seqüencial, només podíem contemplar els casos:
  - $\mathbf{K_i} = \mathbf{K}$ (trobat, cerca acabada),
  - i $\mathbf{K_i} \neq \mathbf{K}$ (element encara no trobat)
- En canvi, si les dades tenen un ordre implícit, contemplarem els casos:
  - $\mathbf{K_i} < \mathbf{K}$ (descartem els registres anteriors a $\mathbf{K_i}$)
  - o bé $\mathbf{K_i} = \mathbf{K}$ (trobat, cerca acabada)
  - o bé $\mathbf{K_i} > \mathbf{K}$ (descartem els registres posteriors a $\mathbf{K_i}$)

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

Busquem: el 5

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

Busquem: el 5

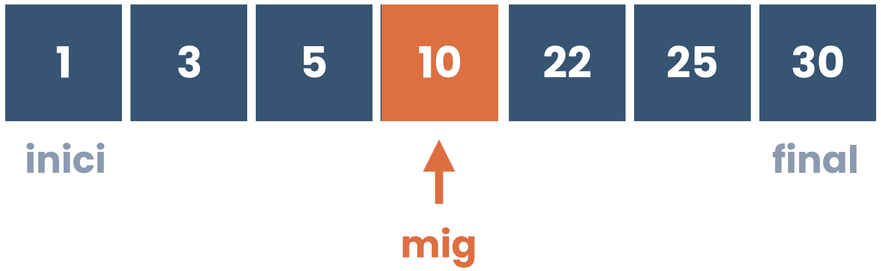
Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

Busquem: el 5



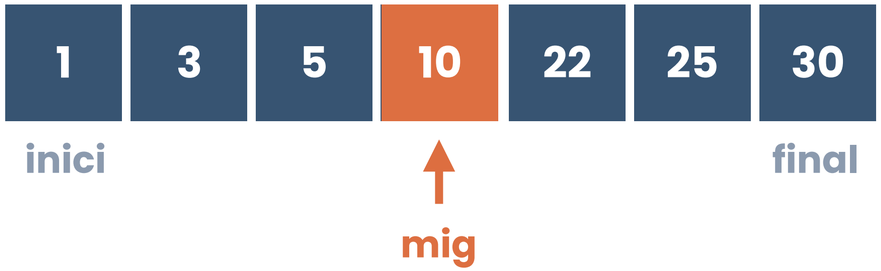
Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

Busquem: el 5



10 > 5. Descartem la part superior

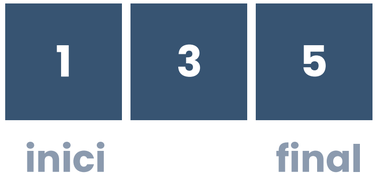
Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

Busquem: el 5



Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

Busquem: el 5

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

Busquem: el 5

3 < 5. Descartem la part inferior

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

Busquem: el 5

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

Busquem: el 5

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

Busquem: el 5

5 = 5. Trobat!

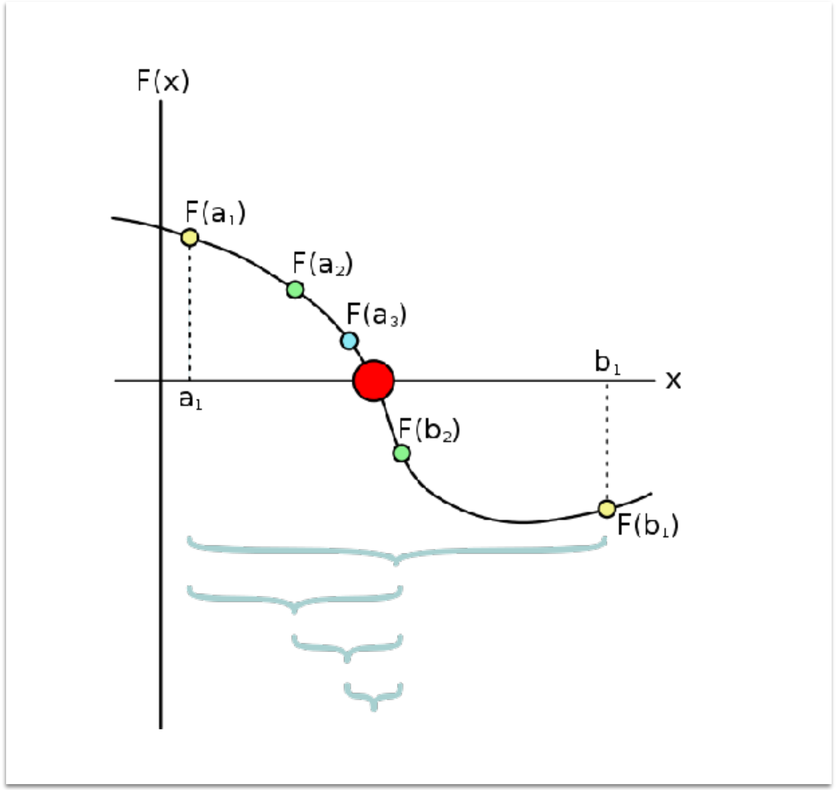
Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

Relacionat: el mètode de bisecció



HTTPS://EN.WIKIPEDIA.ORG/WIKI/BISECTION_METHOD

Descripció del mètode:

- Es comprova que $f(a) \cdot f(b) < 0$
- Es calcula el punt mitjà *m* de l'interval *[a,b]* i s'avalua *f(m)*.
- Si *f(m)=0*, *m* és una arrel. Si no, es comprova que *f(m)* té signe contrari que *f(a)* ó *f(b)*.
- Es redefineix l'interval *[a,b]* com *[a,m]* ó *[m,b]* segons s'haja determinat en quin d'aquests intervals es produeix un canvi de signe.
- Es repeteix el procés amb l'interval fins a arribar a la precisió desitjada.

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Requereix que el conjunt de registres estigui ordenat.
- Començar a cercar pel valor a la meitat de la taula. Això ens determina quina meitat de la taula explorarem a continuació i quina descartarem. Amb la meitat que ens quedem, repetim aquest procés.

- Pseudocodi:

```
$ Suposem ordenació creixent
funció cercaBinaria (vector: taula[] d'enter,
mida: enter, clau: enter) retorna booleà és
var
 ini, fi, mig: enter;
 trobat: booleà;
fvar
inici
   ini := 0;
   fi := mida-1;
   trobat := fals;
  mentre (ini <= fi i no(trobat)) fer
     mig := ((ini+fi) div 2);
     si (vector[mig] < clau)
        $ Descartem la meitat inferior
        ini := mig + 1;
     sino si (vector[mig] > clau)
         $ Descartem la meitat superior
         fi := mig - 1;
      sino $ vector[mig] == clau
        trobat := cert; $ Trobat a la posicio "mig"
      fsi
  fmentre
 retorna (trobat);
ffunció
```

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

- Variacions:
- Si hi ha valors repetits, aquest algorisme no assegura quin dels valors repetits retornarà
  - Variació per retornar sempre l'element esquerre
  - Variació per retornar sempre l'element dret
  - ....

- Curiositats:
- En vectors molt grans, la suma ini+fi, que necessitem per calcular (ini+fi)/2 pot donar un overflow del tipus que utilitzem. Per evitar-ho, es pot calcular com: ini + (fi-ini)/2

- Pseudocodi:

```
$ Suposem ordenació creixent
funció cercaBinaria (vector: taula[] d'enter,
mida: enter, clau: enter) retorna booleà és
var
 ini, fi, mig: enter;
 trobat: booleà;
fvar
inici
  ini := 0;
  fi := mida-1;
  trobat := fals;
 mentre (ini <= fi i no(trobat)) fer
   mig := ((ini+fi) div 2);
   si (vector[mig] < clau)
     $ Descartem la meitat inferior
     ini := mig + 1;
   sino si (vector[mig] > clau)
      $ Descartem la meitat superior
      fi := mig - 1;
    sino $ vector[mig] == clau
      trobat := cert; $ Trobat a la posicio "mig"
    fsi
 fmentre
 retorna (trobat);
ffunció
```

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

Anàlisi de costos:

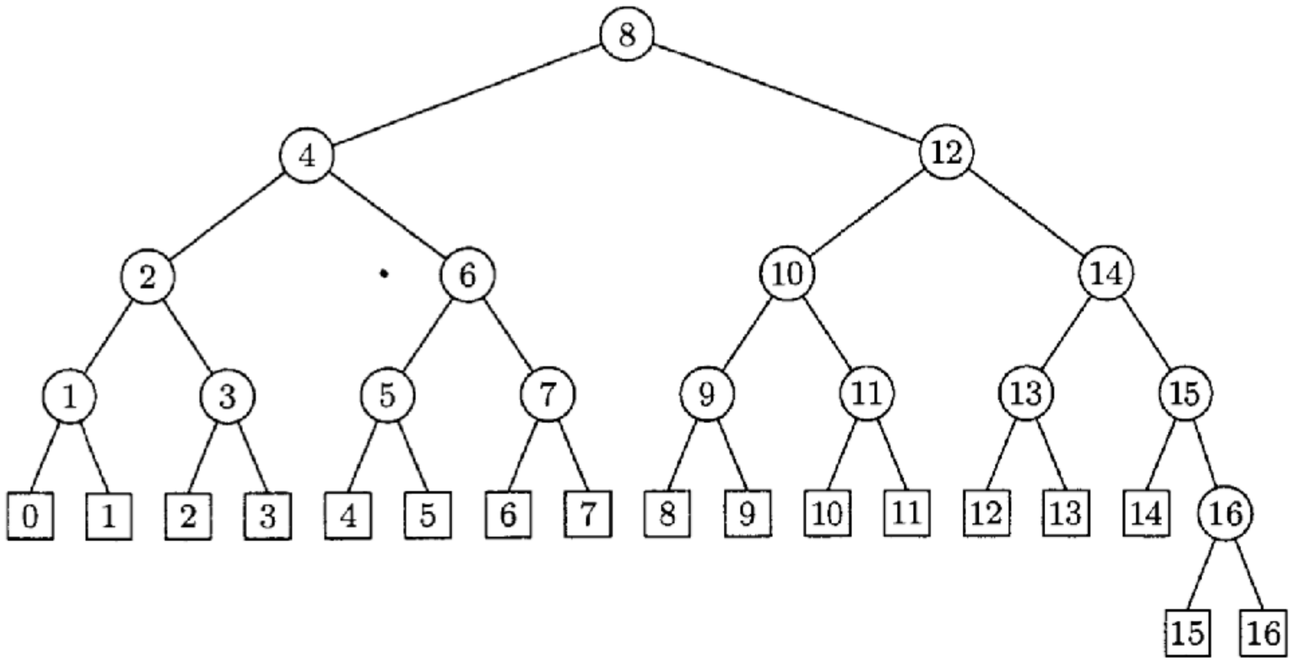


Fig. 5. A comparison tree that corresponds to binary search when $N = 16$.

SOURCE: KNUTH VOL. 3 PAGE 412

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

Anàlisi de costos:

- En el **millor cas**, l'element està a la posició del mig. Només cal una iteració.

  E.g. busquem el '8'

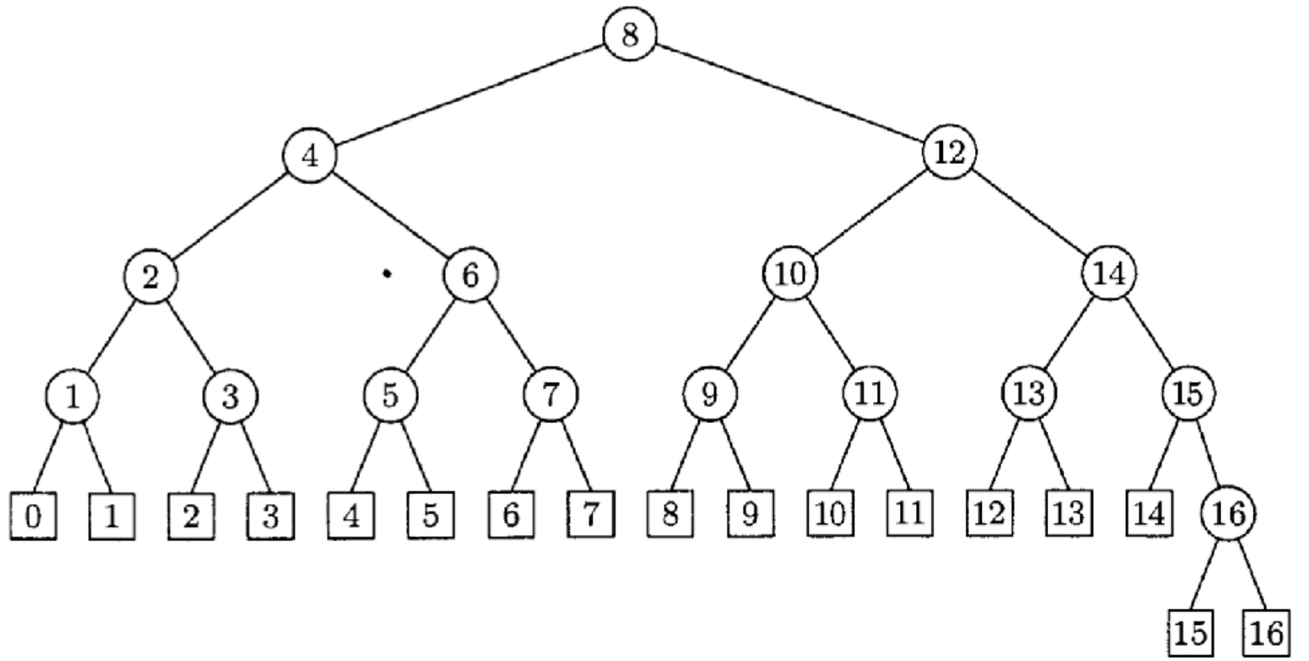


Fig. 5. A comparison tree that corresponds to binary search when $N = 16$.

SOURCE: KNUTH VOL. 3 PAGE 412

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

Anàlisi de costos:

- En el **millor cas**, l'element està a la posició del mig. Només cal una iteració.

  E.g. busquem el '8' **Best case = O(1)**

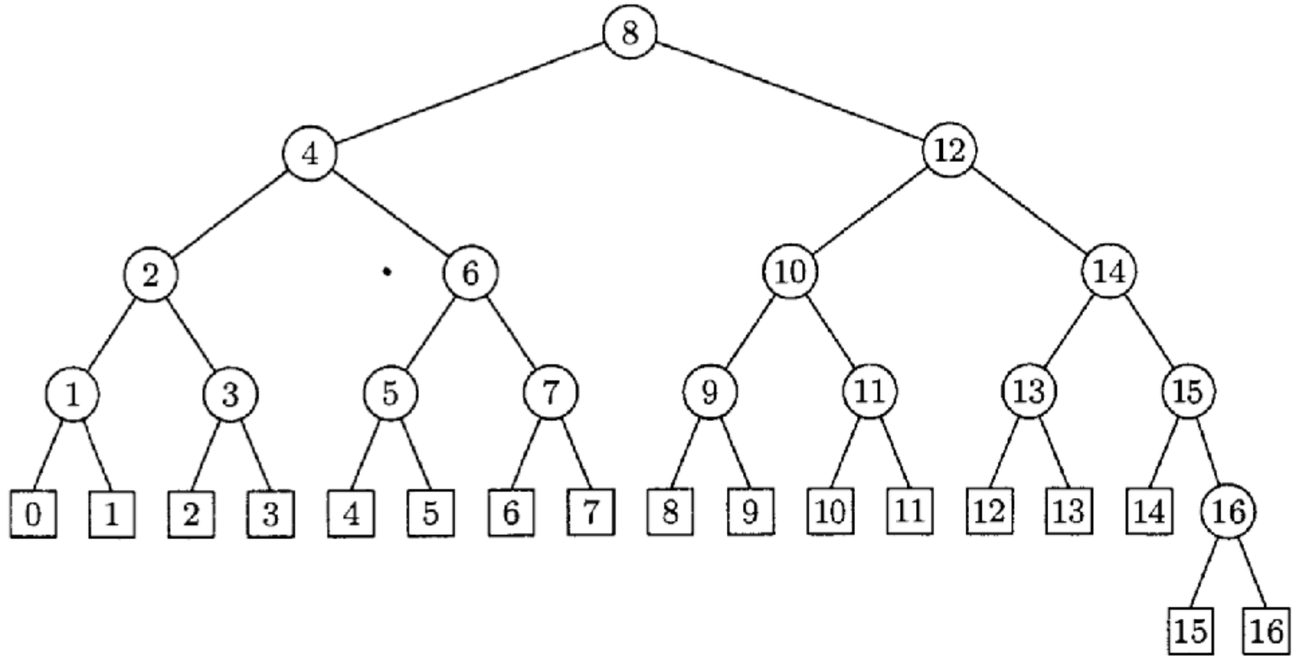


Fig. 5. A comparison tree that corresponds to binary search when $N = 16$.

SOURCE: KNUTH VOL. 3 PAGE 412

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

Anàlisi de costos:

- En el **millor cas**, l'element està a la posició del mig. Només cal una iteració.

  E.g. busquem el '8' **Best case = O(1)**

- En el **pitjor cas**, l'element està al nivell més profund de l'arbre. Quantes iteracions es fan?

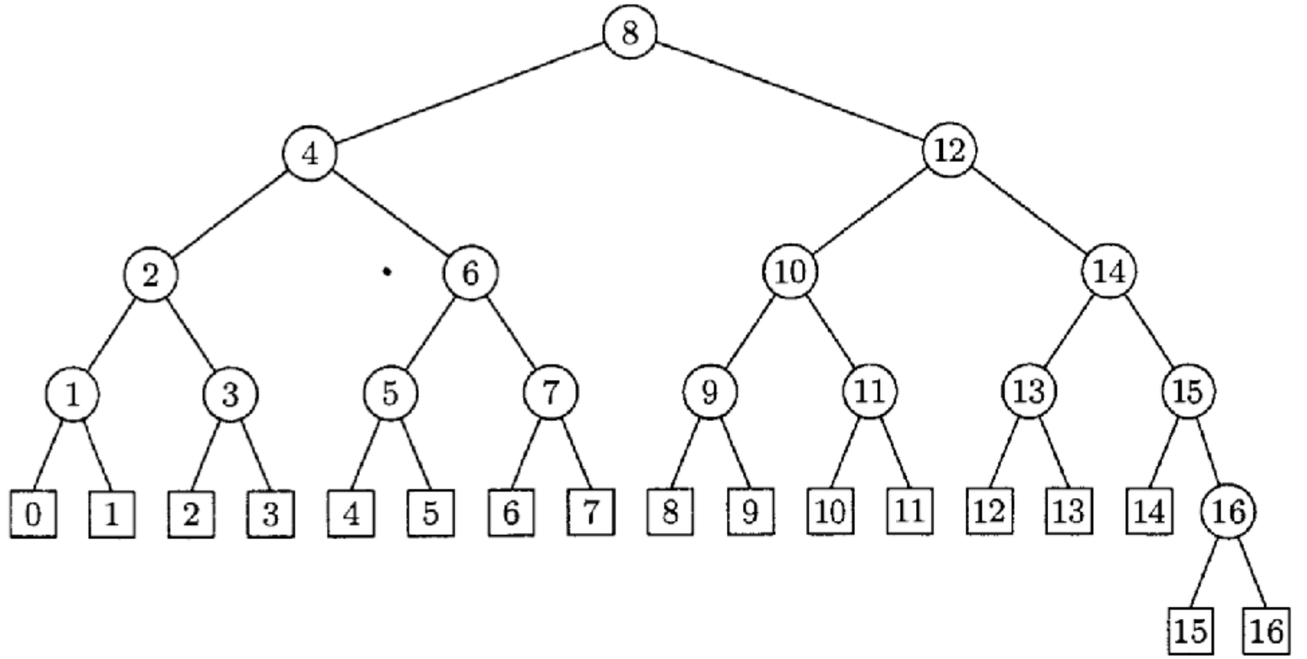


Fig. 5. A comparison tree that corresponds to binary search when $N = 16$.

SOURCE: KNUTH VOL. 3 PAGE 412

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

Anàlisi de costos:

- En el **millor cas**, l'element està a la posició del mig. Només cal una iteració.

  E.g. busquem el '8' **Best case = O(1)**

- En el **pitjor cas**, l'element està al nivell més profund de l'arbre. Quantes iteracions es fan?

Busquem: el 5

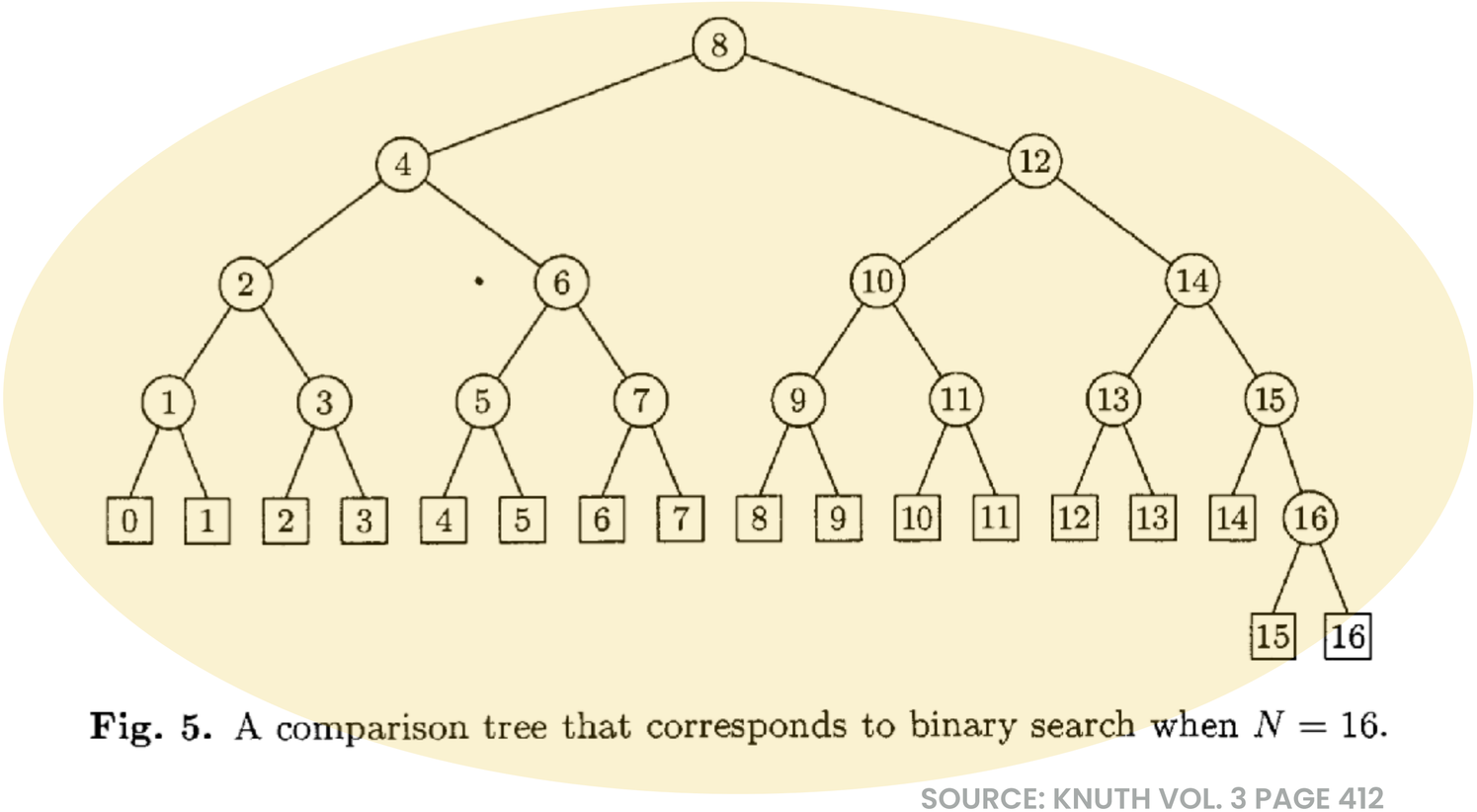


Fig. 5. A comparison tree that corresponds to binary search when $N = 16$.

SOURCE: KNUTH VOL. 3 PAGE 412

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

Anàlisi de costos:

- En el **millor cas**, l'element està a la posició del mig. Només cal una iteració.

  E.g. busquem el '8' **Best case = O(1)**

- En el **pitjor cas**, l'element està al nivell més profund de l'arbre. Quantes iteracions es fan?

Busquem: el 5

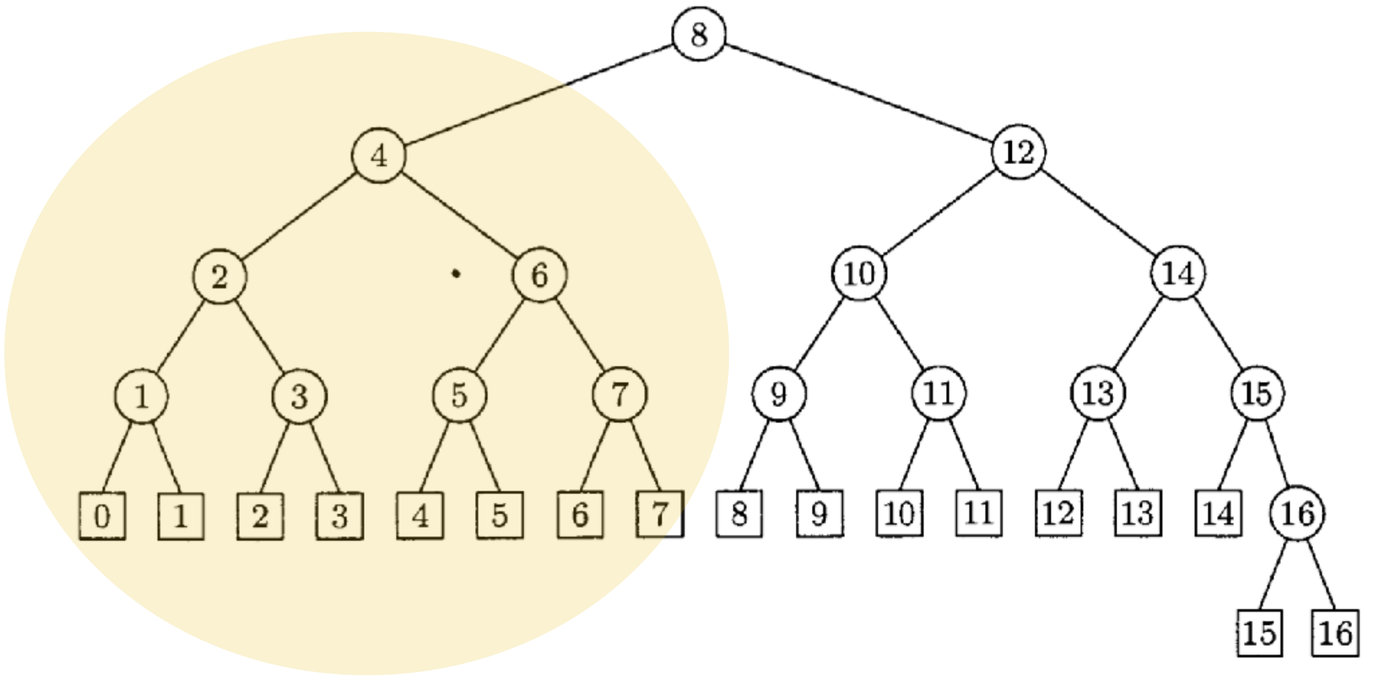


Fig. 5. A comparison tree that corresponds to binary search when $N = 16$.

SOURCE: KNUTH VOL. 3 PAGE 412

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

Anàlisi de costos:

- En el **millor cas**, l'element està a la posició del mig. Només cal una iteració.

  E.g. busquem el '8' **Best case = O(1)**

- En el **pitjor cas**, l'element està al nivell més profund de l'arbre. Quantes iteracions es fan?

Busquem: el 5

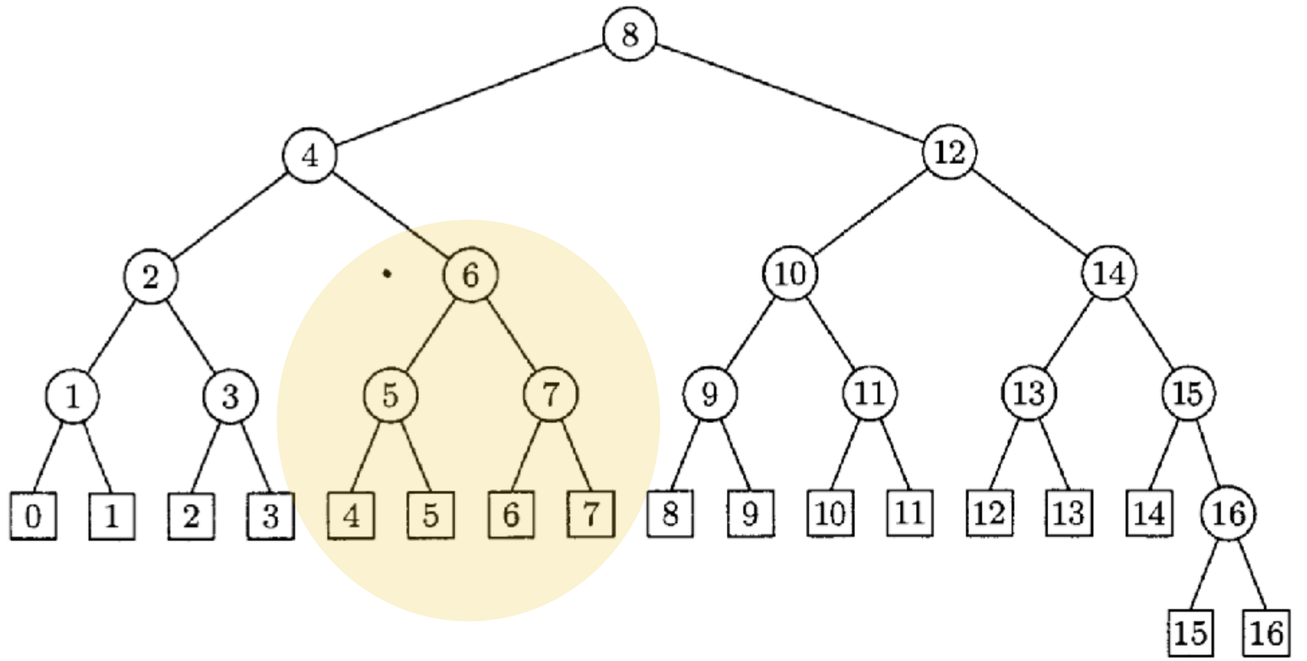


Fig. 5. A comparison tree that corresponds to binary search when $N = 16$.

SOURCE: KNUTH VOL. 3 PAGE 412

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

Anàlisi de costos:

- En el **millor cas**, l'element està a la posició del mig. Només cal una iteració.

  E.g. busquem el '8' **Best case = O(1)**

- En el **pitjor cas**, l'element està al nivell més profund de l'arbre. Quantes iteracions es fan?

Busquem: el 5

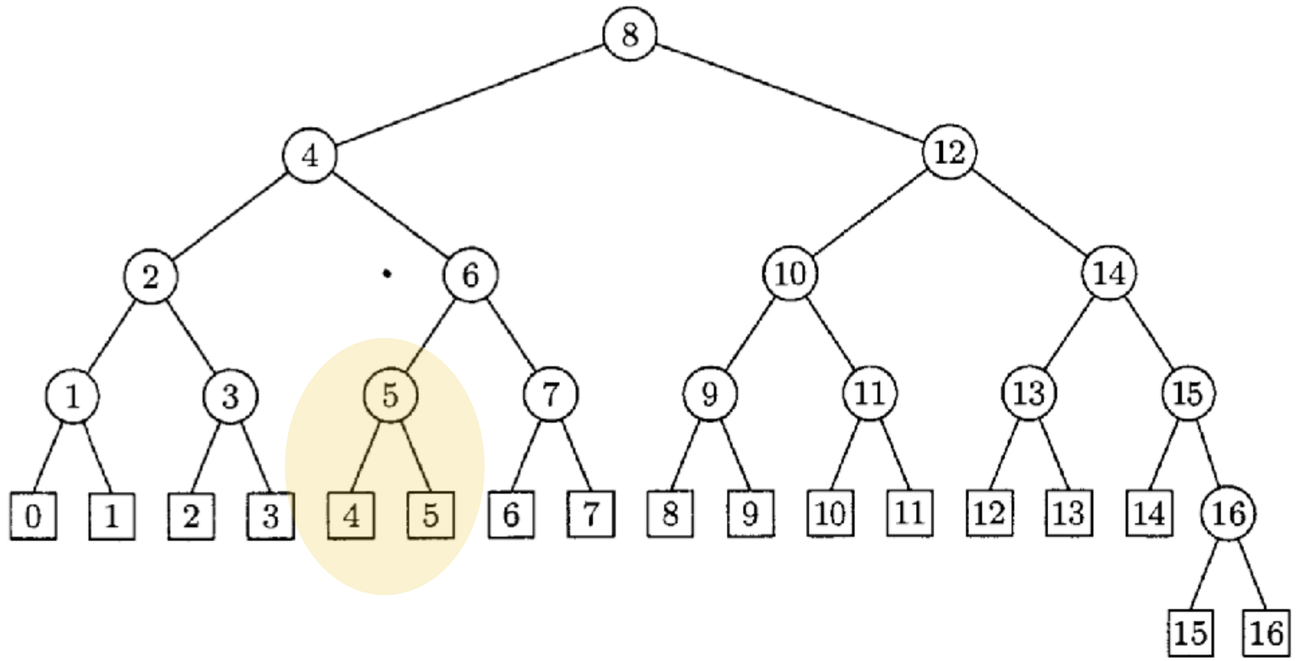


Fig. 5. A comparison tree that corresponds to binary search when $N = 16$.

SOURCE: KNUTH VOL. 3 PAGE 412

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

Anàlisi de costos:

- En el **millor cas**, l'element està a la posició del mig. Només cal una iteració.

  E.g. busquem el '8' **Best case = O(1)**

- En el **pitjor cas**, l'element està al nivell més profund de l'arbre. Quantes iteracions es fan?

Busquem: el 5

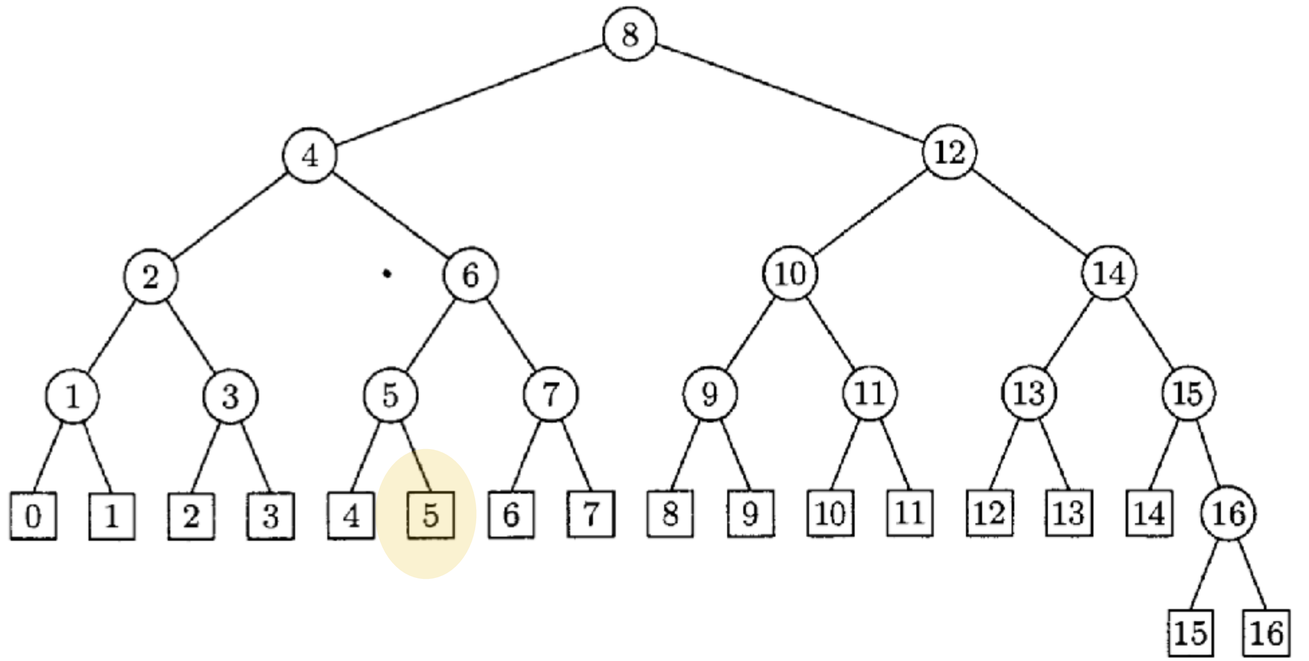


Fig. 5. A comparison tree that corresponds to binary search when $N = 16$.

SOURCE: KNUTH VOL. 3 PAGE 412

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

Anàlisi de costos:

- En el **millor cas**, l'element està a la posició del mig. Només cal una iteració.

  E.g. busquem el '8' **Best case = O(1)**

- En el **pitjor cas**, l'element està al nivell més profund de l'arbre. Quantes iteracions es fan?

Busquem: el 5

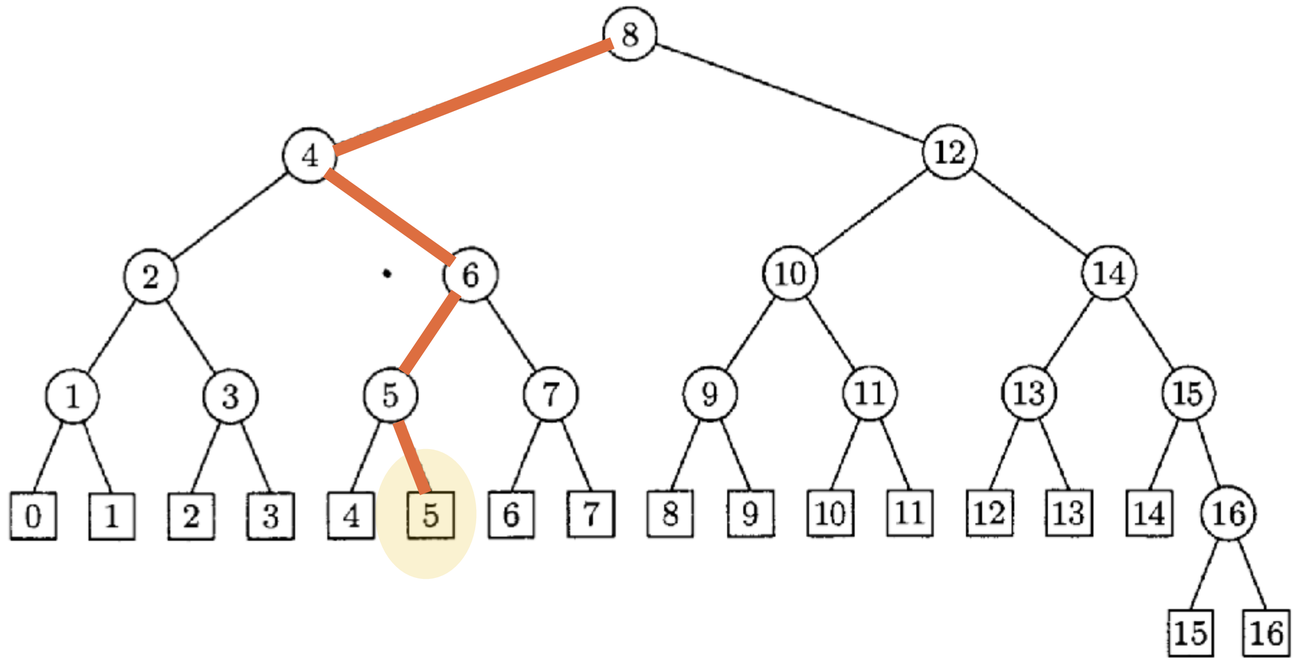


Fig. 5. A comparison tree that corresponds to binary search when $N = 16$.

SOURCE: KNUTH VOL. 3 PAGE 412

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

Anàlisi de costos:

- En el **millor cas**, l'element està a la posició del mig. Només cal una iteració.

  E.g. busquem el '8' **Best case = O(1)**

- En el **pitjor cas**, l'element està al nivell més profund de l'arbre. Quantes iteracions es fan?

  Worst case = O(log n)

Busquem: el 5

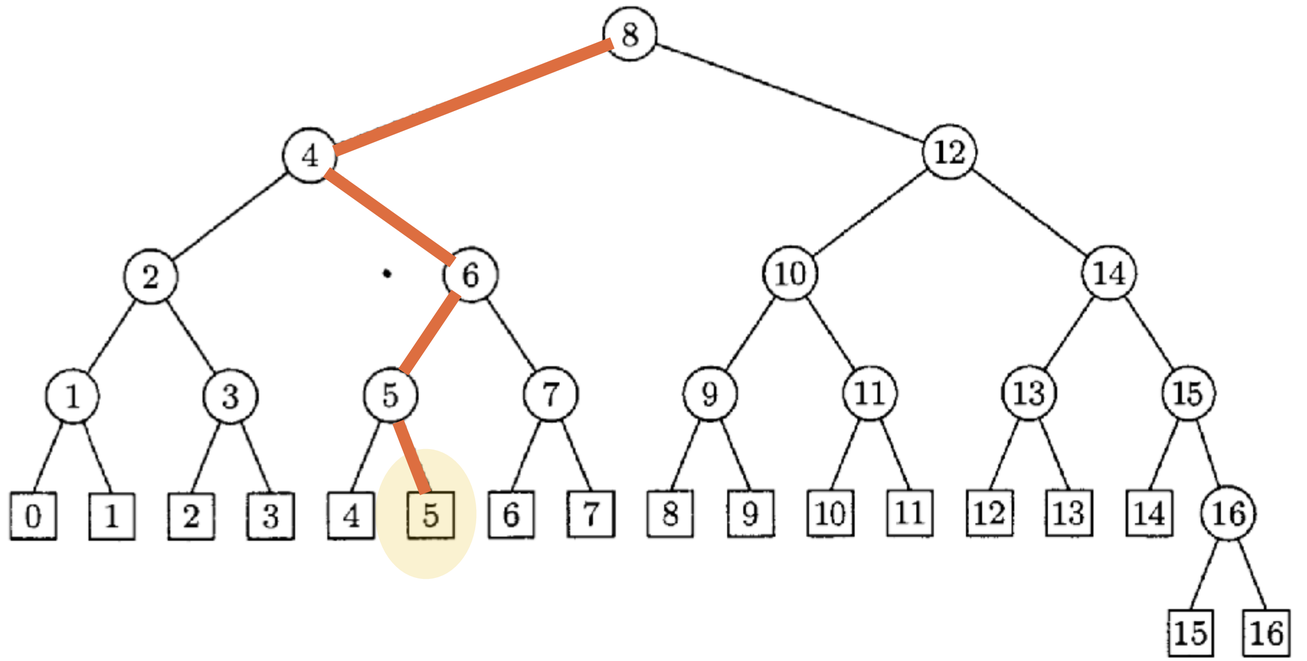


Fig. 5. A comparison tree that corresponds to binary search when $N = 16$.

SOURCE: KNUTH VOL. 3 PAGE 412

Cerca

Cerca en conjunts ordenats:

Cerca binària / dicotòmica / bisecció

Anàlisi de costos:

- En el **millor cas**, l'element està a la posició del mig. Només cal una iteració.

  E.g. busquem el '8' **Best case = O(1)**

- En el **pitjor cas**, l'element està al nivell més profund de l'arbre. Quantes iteracions es fan?

  Worst case = O(log n)

- En el **cas mig**, s'han de tenir en compte els casos exitosos vs. no exitosos. Càlcul aquí:

  https://en.wikipedia.org/wiki/Binary_search_algorithm#Derivation_of_average_case

  Average case = O(log n)

Busquem: el 5

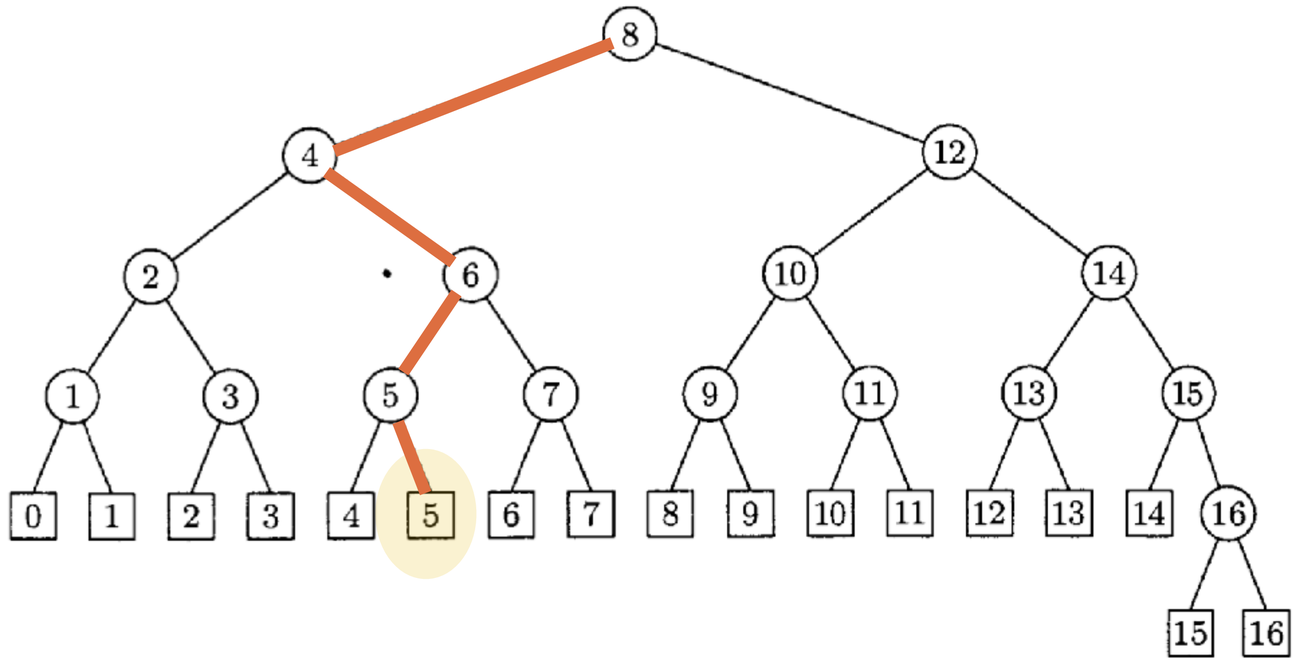


Fig. 5. A comparison tree that corresponds to binary search when $N = 16$.

SOURCE: KNUTH VOL. 3 PAGE 412

ORDENACIÓ

Ordenació

Necessitat i definicions

- Mantenir un conjunt de dades ordenat ens serveix per poder fer cerques més eficients (tot i que no surten a compte si només s'han de fer una vegada)
- Alguns processos matemàtics requereixen de l'ordenació d'elements. E.g. trobar la mediana, els quartils...

SOURCE: WIRTH PAGE 62

Ordenació

Necessitat i definicions

- Mantenir un conjunt de dades ordenat ens serveix per poder fer cerques més eficients (tot i que no surten a compte si només s'han de fer una vegada)
- Alguns processos matemàtics requereixen de l'ordenació d'elements. E.g. trobar la mediana, els quartils...

SOURCE: WIRTH PAGE 62

Cada element és un registre r amb clau k

Definim que tenim els següents registres:

$$r_1, r_2, \ldots, r_n$$

Ordenar consisteix en permutar aquests registres

$$a_{k_1}, a_{k_2}, \ldots, a_{k_n}$$

De manera que donada una funció d'ordenació f es verifiqui:

$$f(a_{k_1}) \leq f(a_{k_2}) \leq \ldots \leq f(a_{k_n})$$

Ordenació

Necessitat i definicions

- Mantenir un conjunt de dades ordenat ens serveix per poder fer cerques més eficients (tot i que no surten a compte si només s'han de fer una vegada)
- Alguns processos matemàtics requereixen de l'ordenació d'elements. E.g. trobar la mediana, els quartils...

Cada element és un registre r amb clau k

Definim que tenim els següents registres:

$$r_1, r_2, \ldots, r_n$$

Ordenar consisteix en permutar aquests registres

$$a_{k_1}, a_{k_2}, \ldots, a_{k_n}$$

De manera que donada una funció d'ordenació f es verifiqui:

$$f(a_{k_1}) \leq f(a_{k_2}) \leq \ldots \leq f(a_{k_n})$$

SOURCE: WIRTH PAGE 62

- Un mètode d'ordenació es denomina **estable** si l'ordre relatiu dels registres amb la mateixa clau no s'altera pel procés d'ordenació

Ordenació

Necessitat i definicions

- Mantenir un conjunt de dades ordenat ens serveix per poder fer cerques més eficients (tot i que no surten a compte si només s'han de fer una vegada)
- Alguns processos matemàtics requereixen de l'ordenació d'elements. E.g. trobar la mediana, els quartils...

SOURCE: WIRTH PAGE 62

Cada element és un registre r amb clau k

Definim que tenim els següents registres:

$$r_1, r_2, \ldots, r_n$$

Ordenar consisteix en permutar aquests registres

$$a_{k_1}, a_{k_2}, \ldots, a_{k_n}$$

De manera que donada una funció d'ordenació f es verifiqui:

$$f(a_{k_1}) \leq f(a_{k_2}) \leq \ldots \leq f(a_{k_n})$$

- Un mètode d'ordenació es denomina **estable** si l'ordre relatiu dels registres amb la mateixa clau no s'altera pel procés d'ordenació

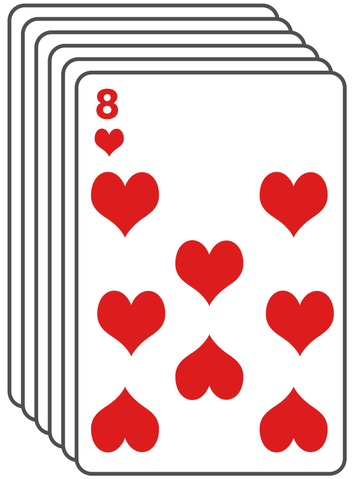
- L'ordenació de taules es realitza **in situ**, és a dir, no es declara una altra taula que s'omple amb els valors ordenats, sinó que només es pot fer servir la memòria que ocupa la nostra taula inicial.

Ordenació

Mètodes directes d'ordenació

- Són els mètodes més bàsics, tot i no ser els més eficients
- Són importants perquè estableixen les bases de molts altres mètodes

SOURCE: CORMEN – INTRODUCTION TO ALGORITHMS



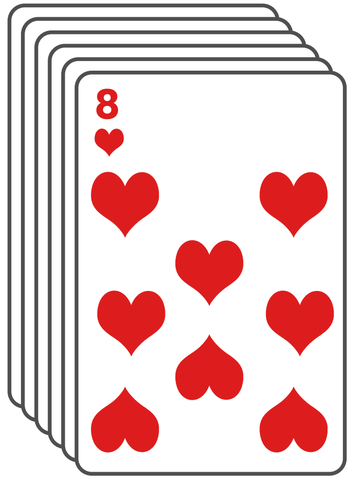
Ordenació

Mètodes directes d'ordenació

- Són els mètodes més bàsics, tot i no ser els més eficients
- Són importants perquè estableixen les bases de molts altres mètodes

(I) Ordenació per inserció

SOURCE: CORMEN – INTRODUCTION TO ALGORITHMS



Ordenació

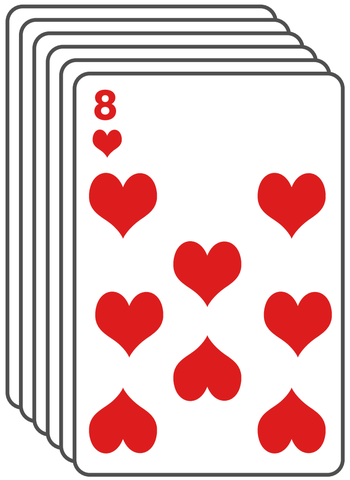
Mètodes directes d'ordenació

- Són els mètodes més bàsics, tot i no ser els més eficients
- Són importants perquè estableixen les bases de molts altres mètodes

SOURCE: CORMEN – INTRODUCTION TO ALGORITHMS

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \ldots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

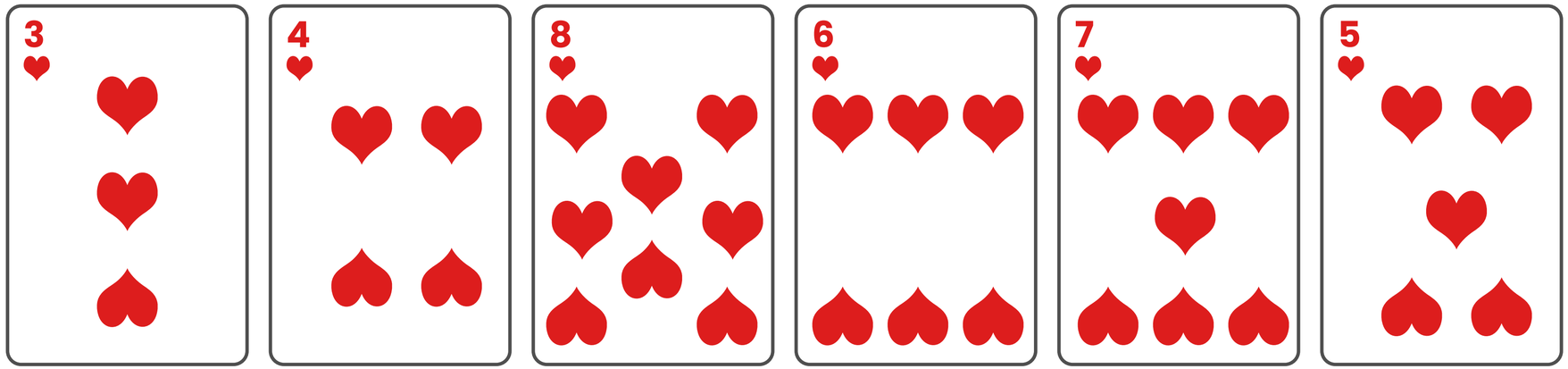


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

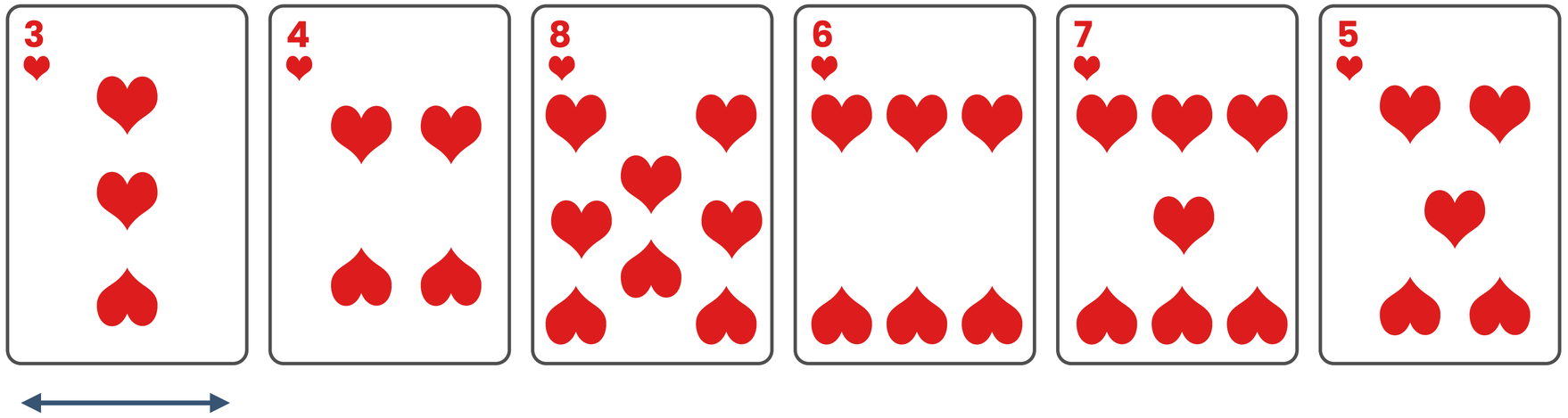


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

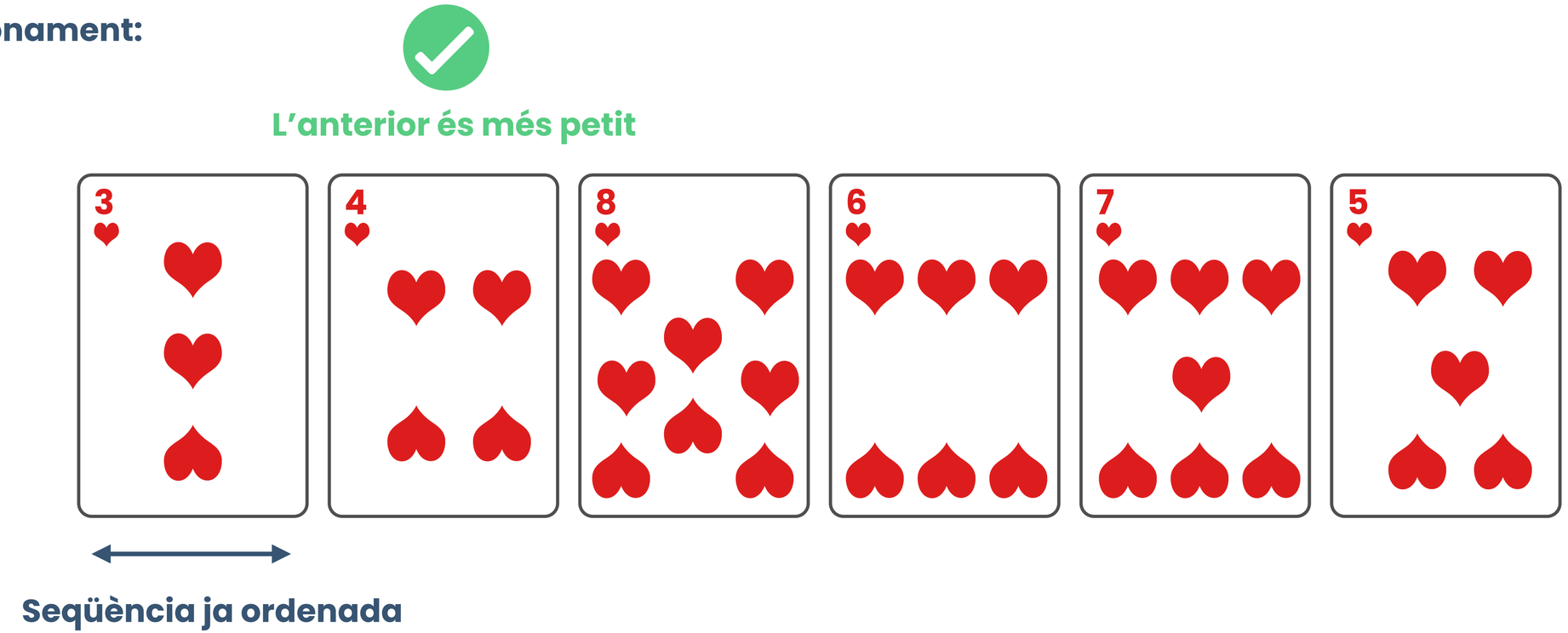


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau K_i, i suposem que la seqüència precedent $K_1, \dots K_{i-1}$ està ja ordenada.
- Inserim cada K_i al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

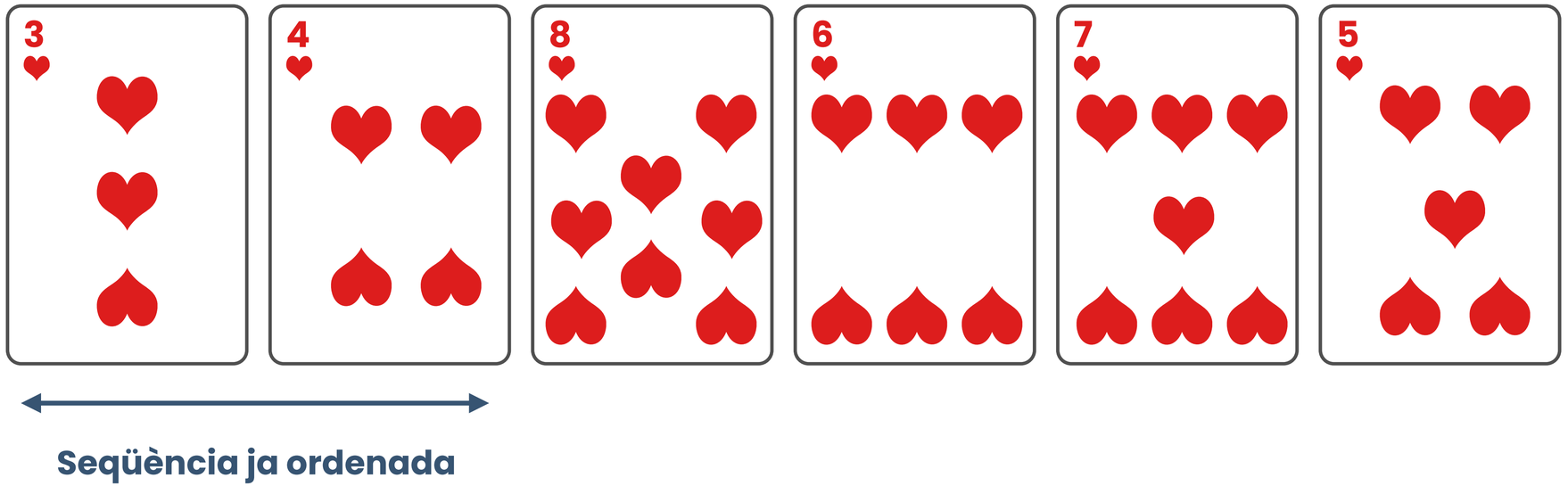


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

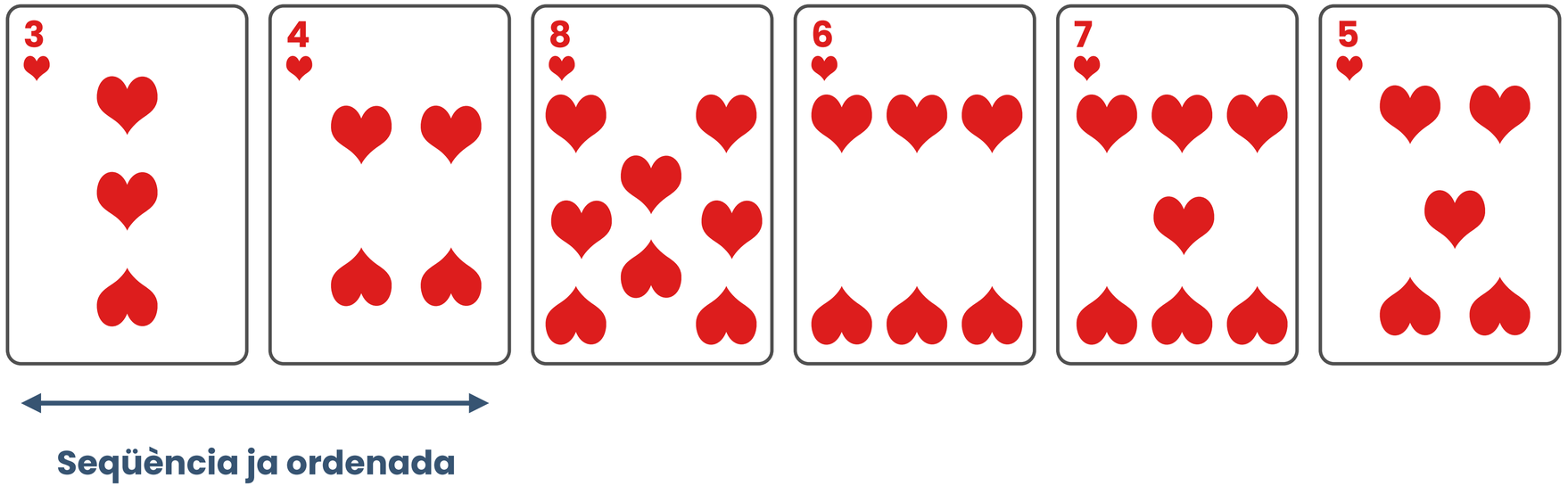


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

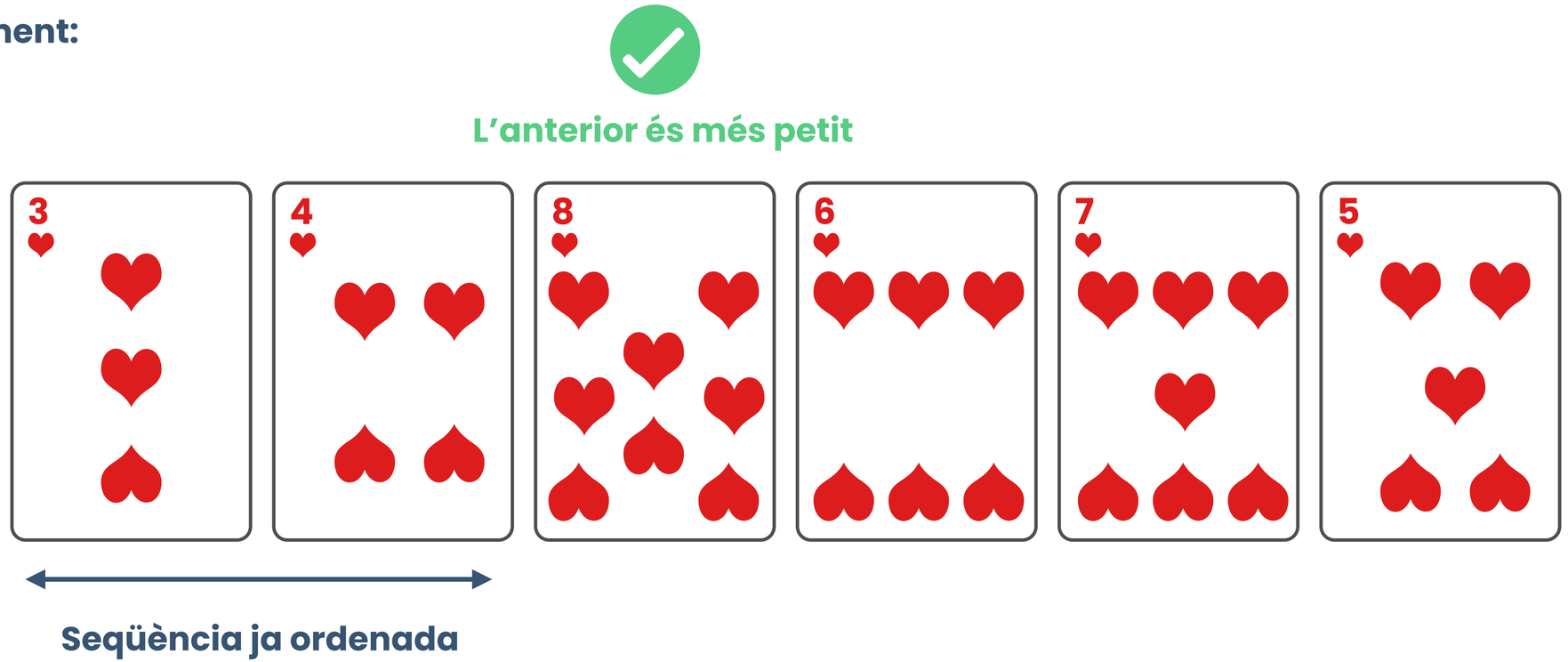


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

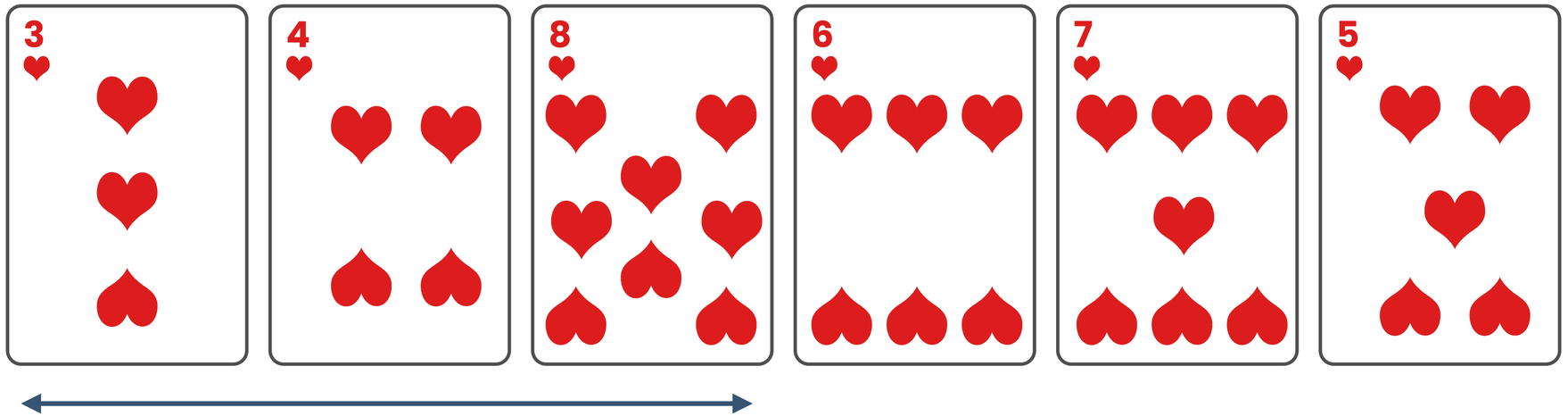


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

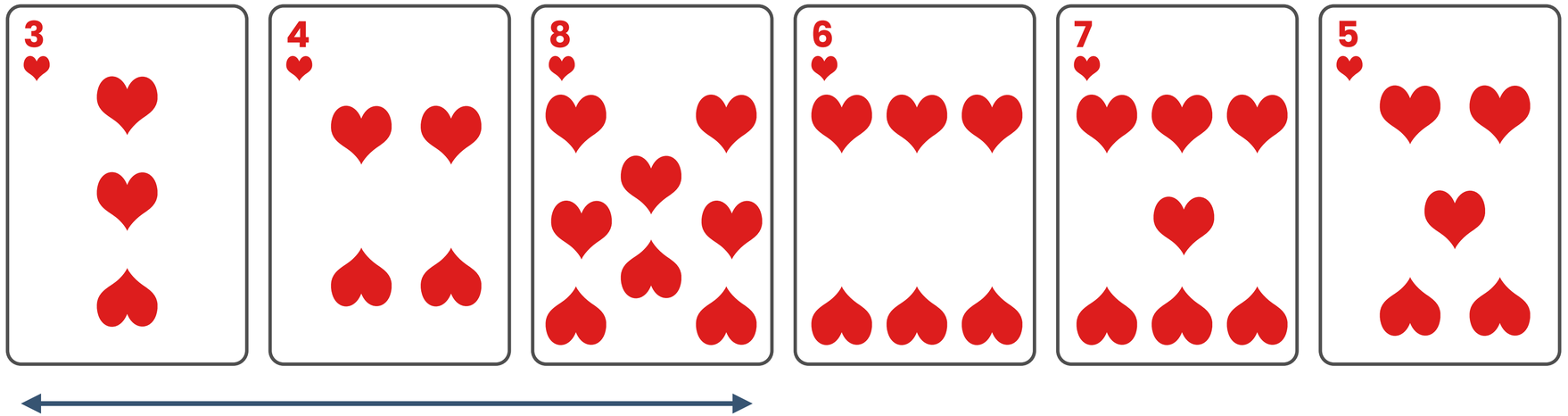


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

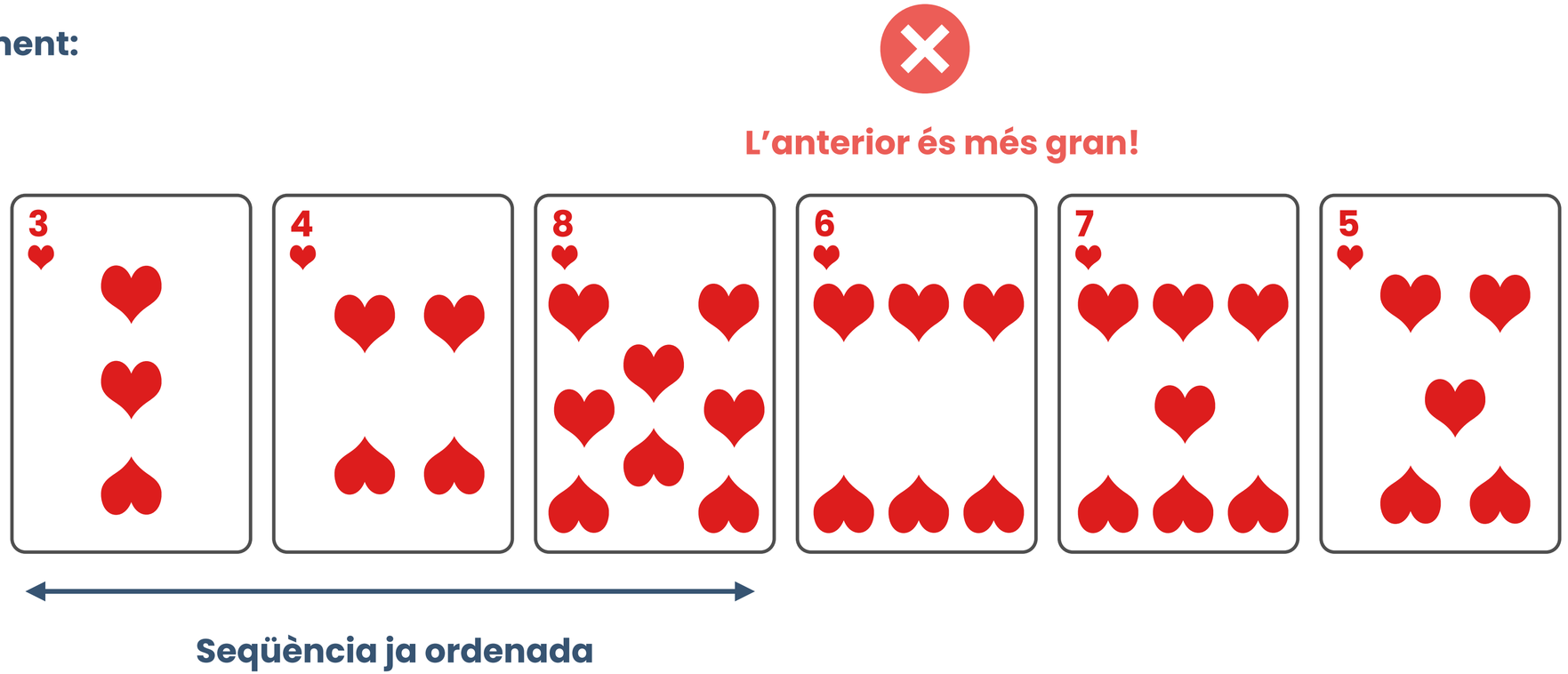


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

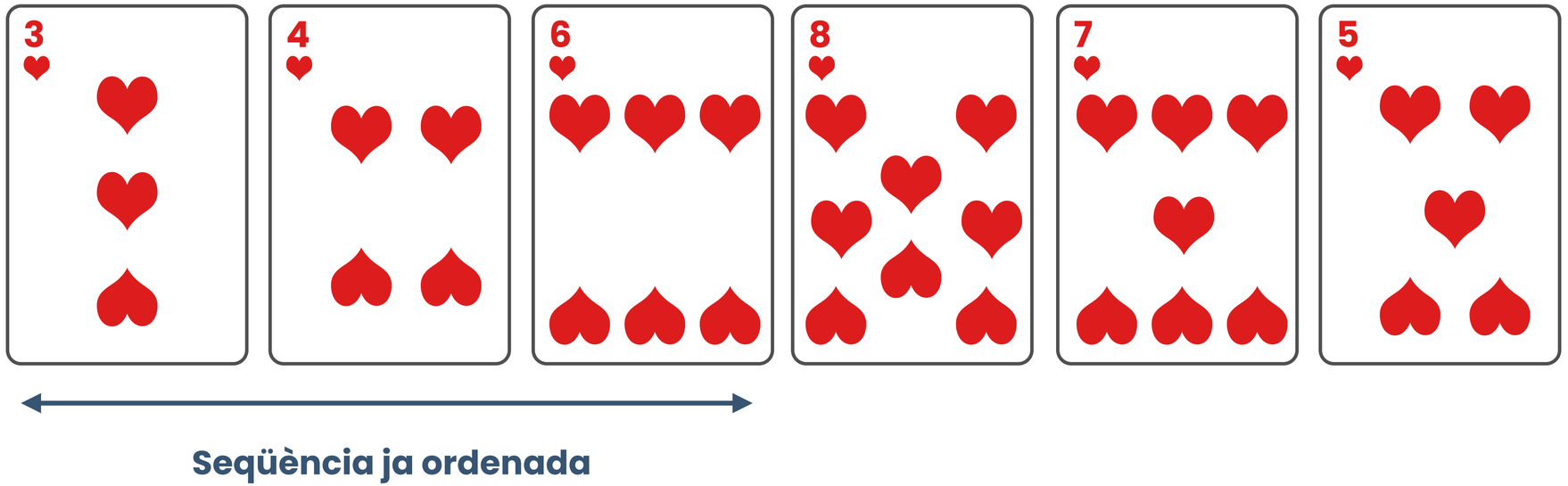


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

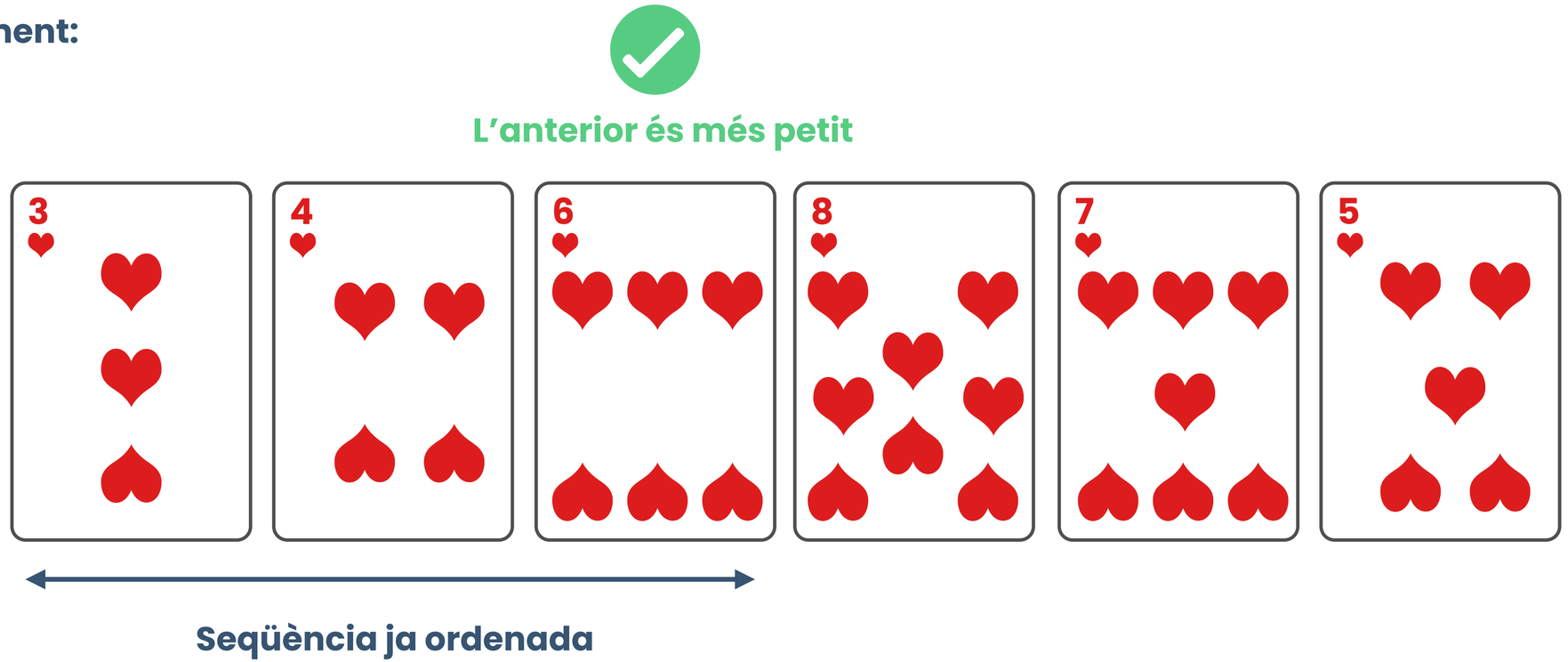


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau K_i, i suposem que la seqüència precedent $K_1, \dots K_{i-1}$ està ja ordenada.
- Inserim cada K_i al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

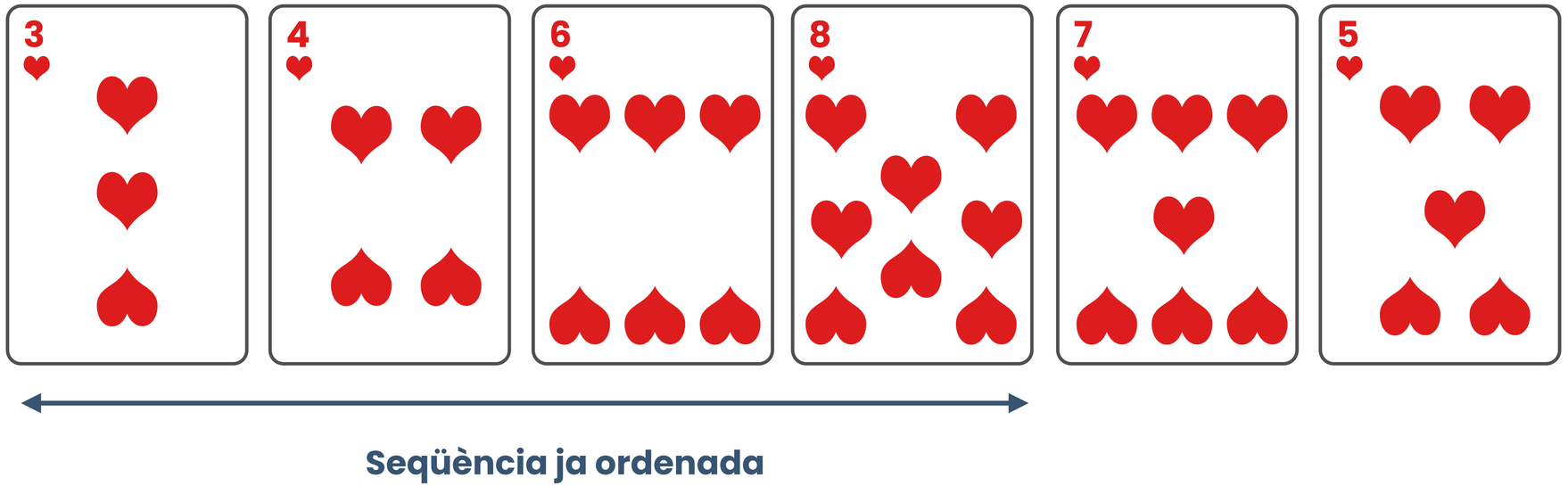


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

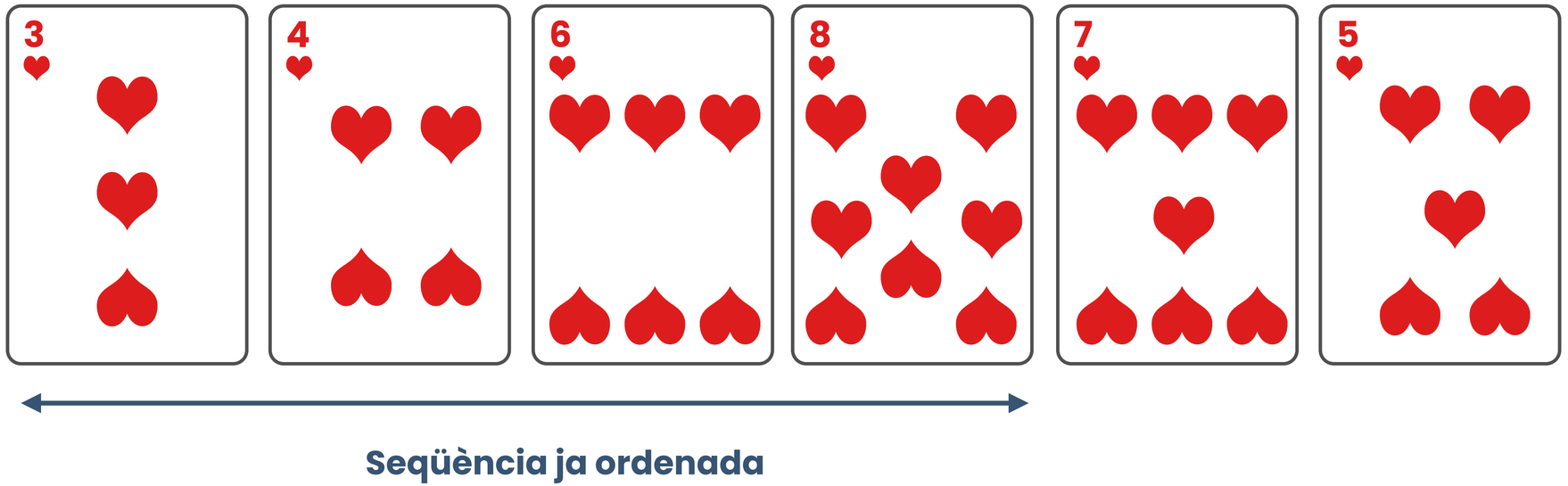


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

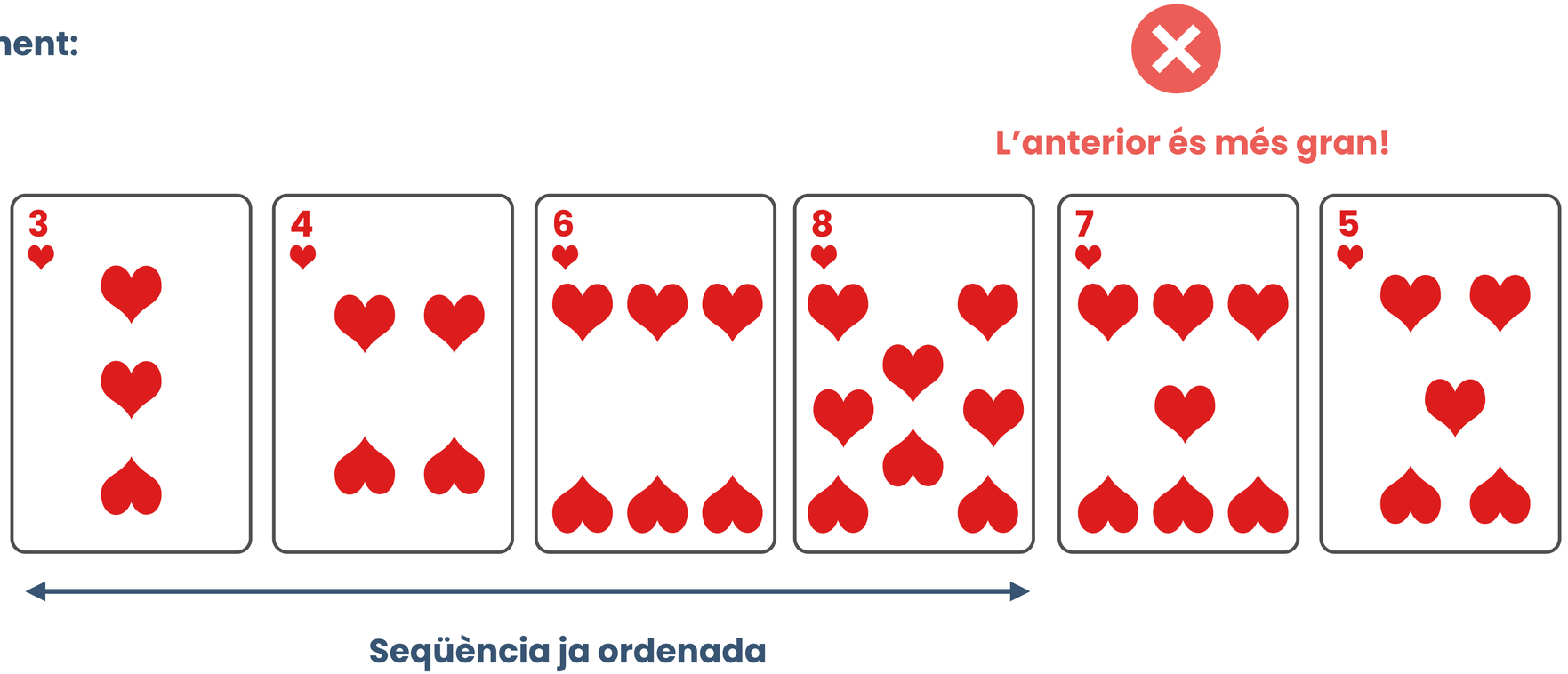


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

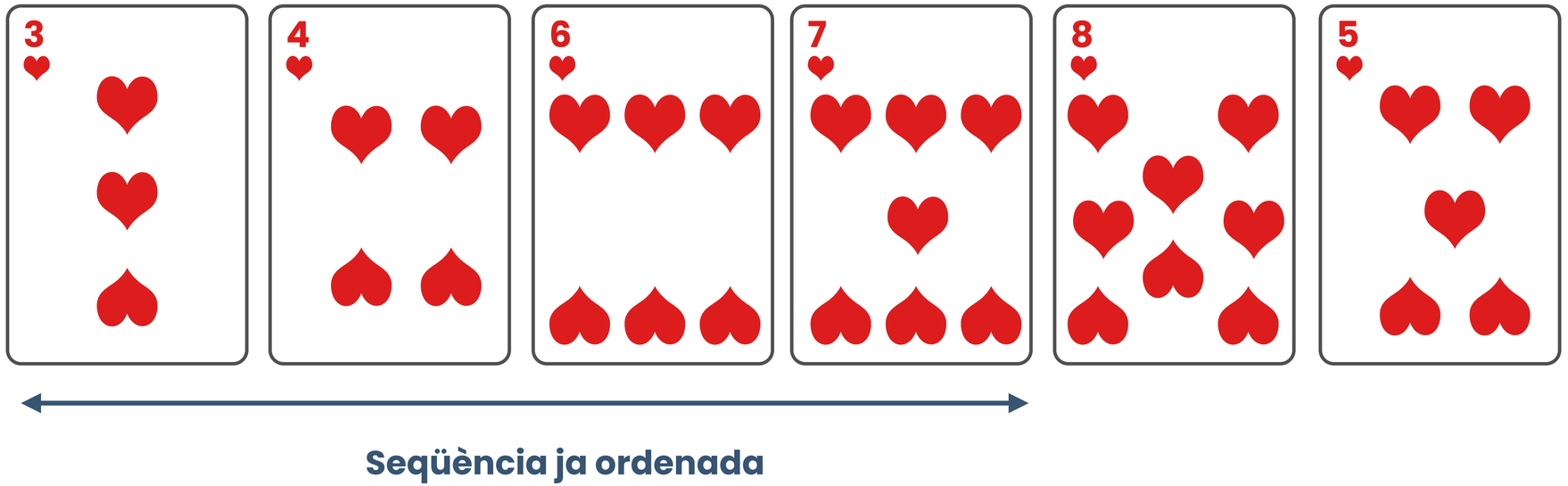


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

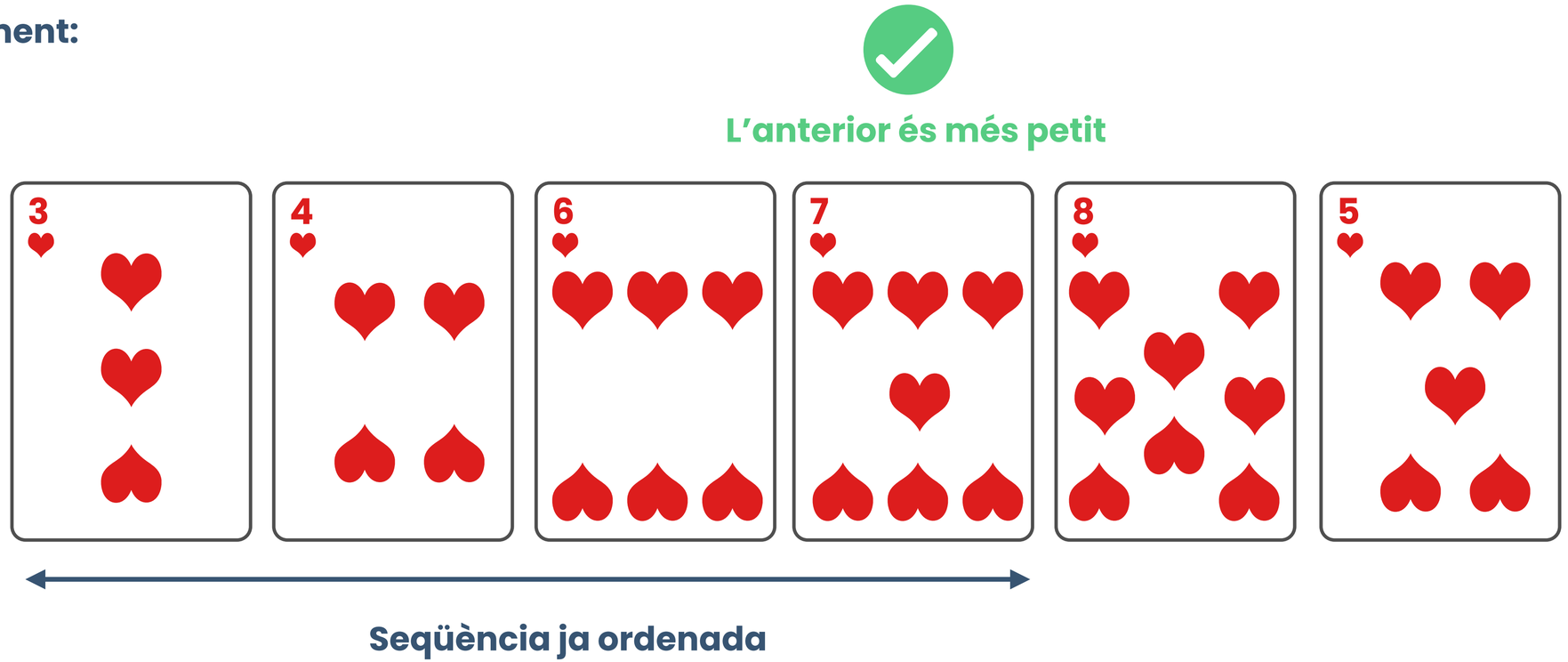


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau K_i, i suposem que la seqüència precedent $K_1, \dots K_{i-1}$ està ja ordenada.
- Inserim cada K_i al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:



Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

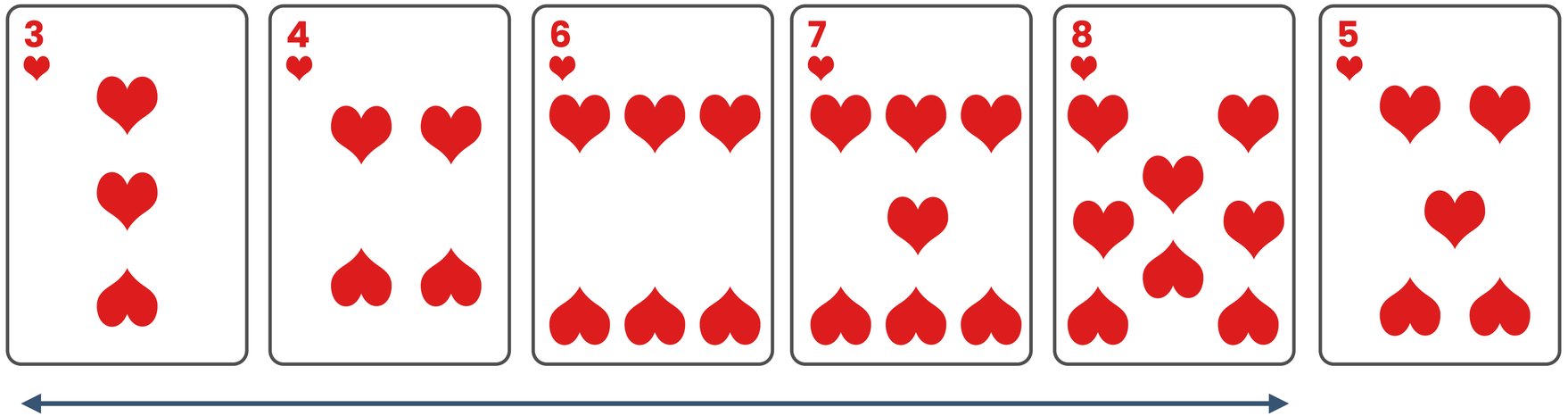
Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

L'anterior és més gran!



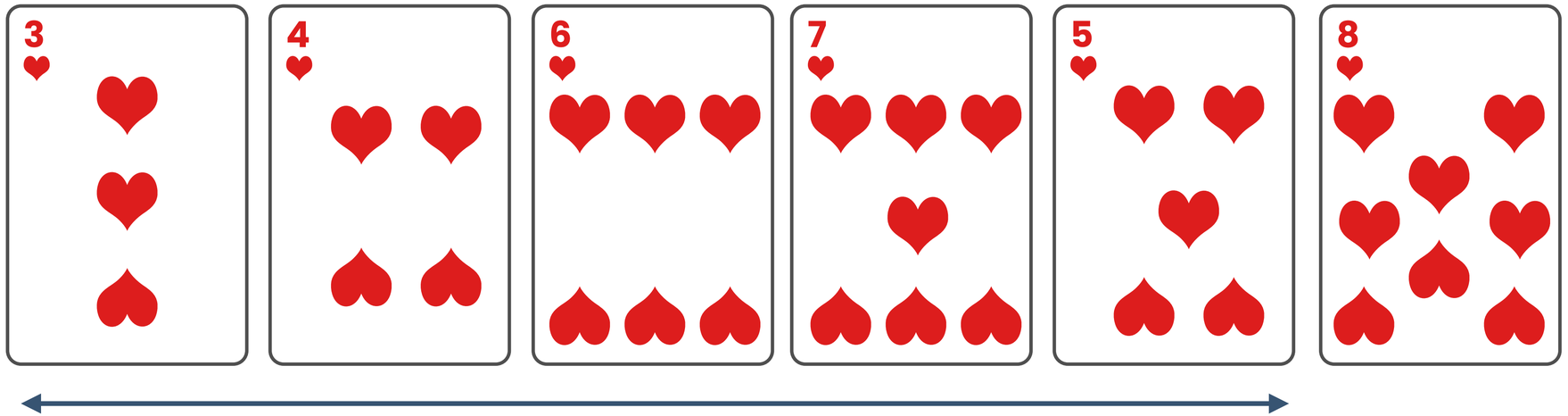
Seqüència ja ordenada

Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

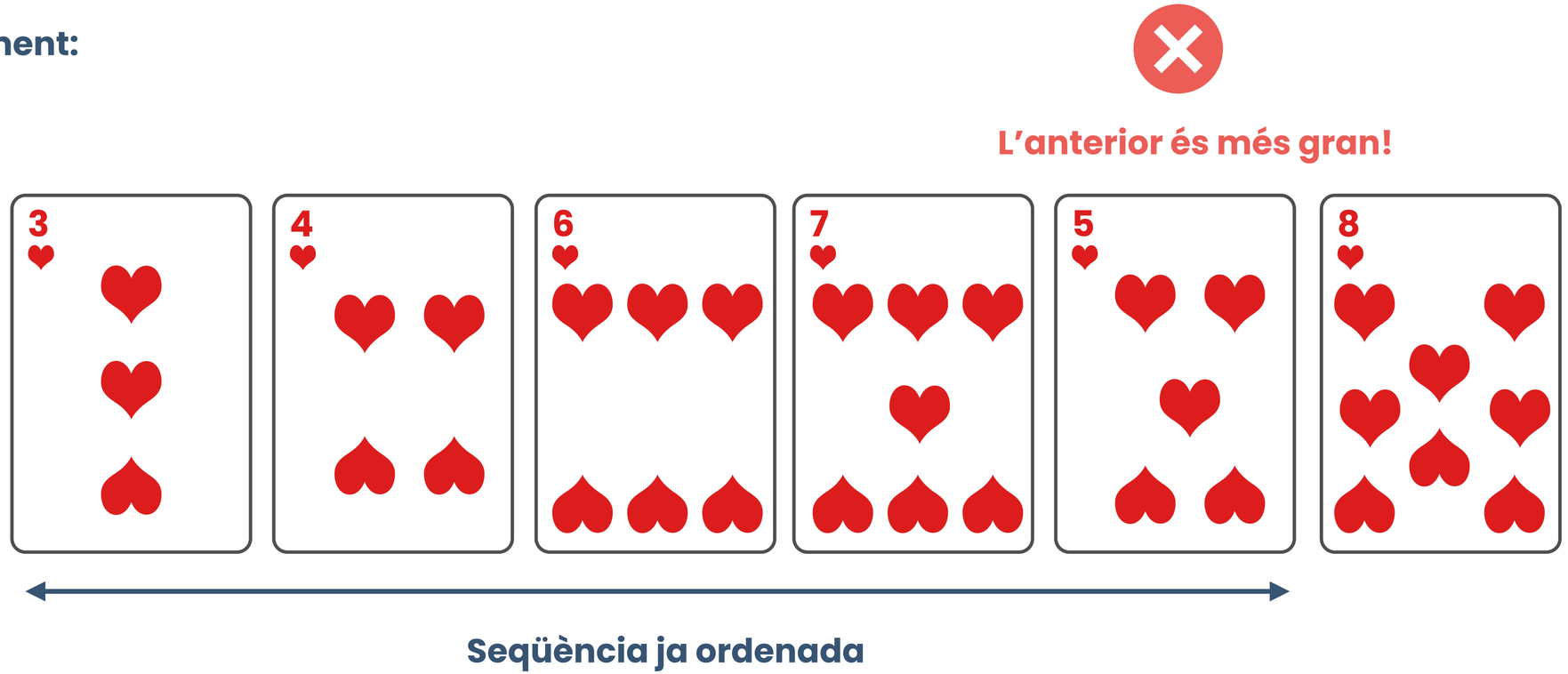


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:

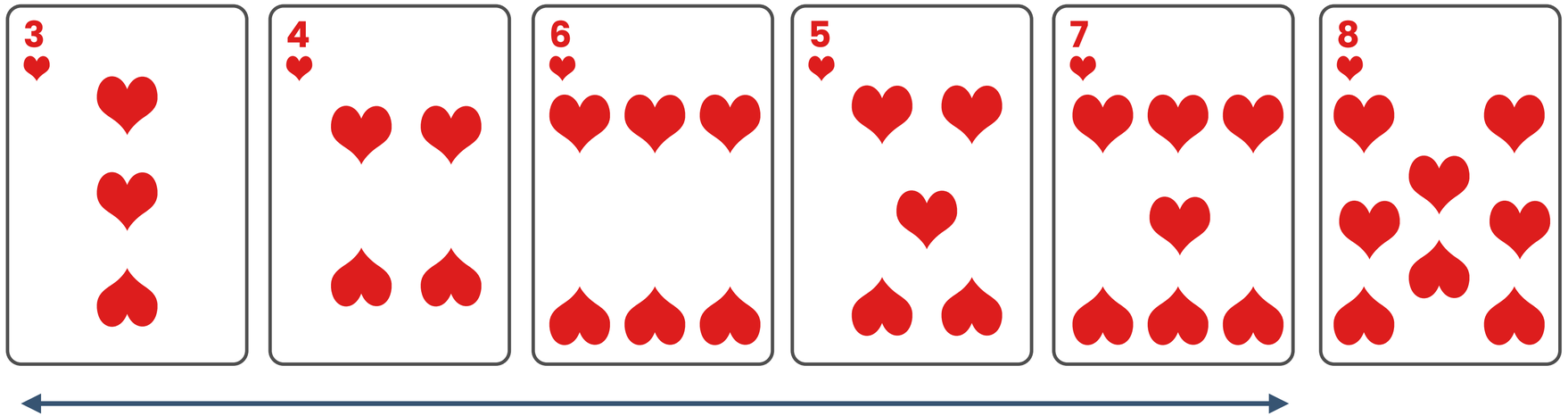


Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:



Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

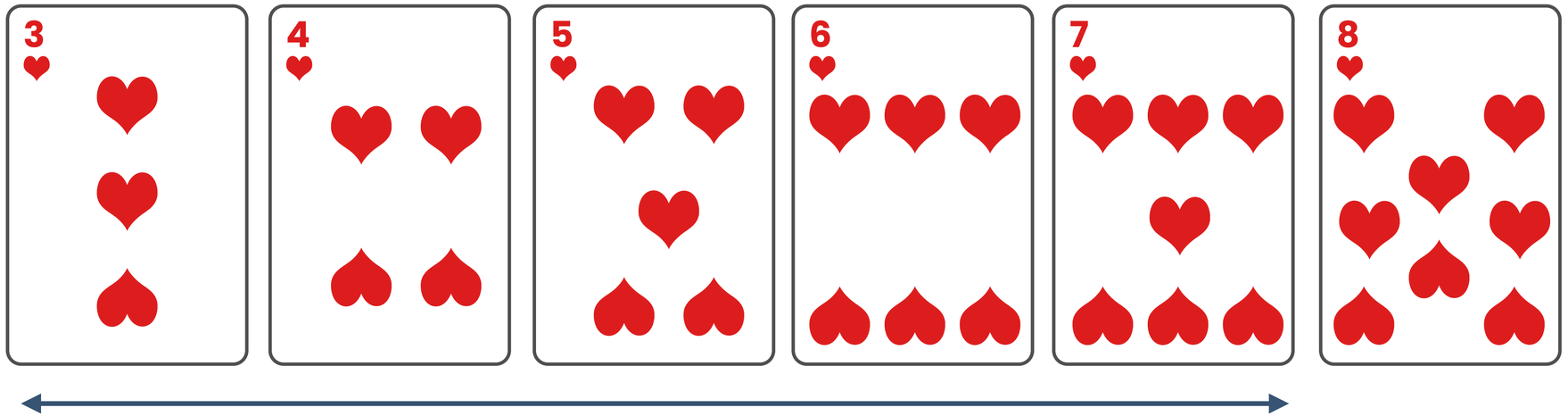
Funcionament:

Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:



Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \dots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

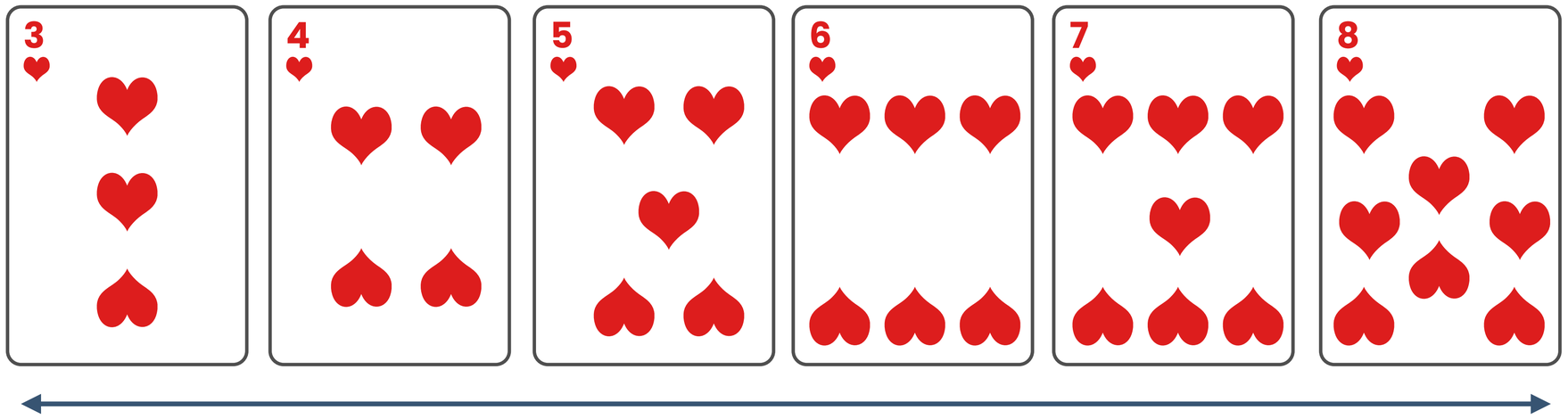
Funcionament:

Ordenació

(I) Ordenació per inserció

- El mètode dels jugadors de cartes
- Partim d'una seqüència de claus inicial i volem arribar a una seqüència de claus ordenada:
- A cada pas, analitzem una clau $\mathbf{K_i}$, i suposem que la seqüència precedent $\mathbf{K_1, \ldots K_{i-1}}$ està ja ordenada.
- Inserim cada $\mathbf{K_i}$ al lloc que li toca dins la seqüència prèviament ordenada (deixant caure la carta fins la posició que li toca).
- Comencem pel segon element perquè el primer no pot anar més a l'esquerra.

Funcionament:



Ordenació

(I) Ordenació per inserció

Algorisme

```
$ Suposem ordenació creixent
acció ordenacio_insercio (v: taula[] d'enter, n_elems:
enter) és
var
 i, j : enter;
 elem : enter;
fvar
inici
    $ Començo a i=1 perquè el primer ja està ordenat
   per (i=1; i < n_elems; i:=i+1) fer
     elem := v[i];
     j := i-1;
     mentre (j >= 0 i elem < v[j]) fer $fer inserció
        v[j+1] = v[j];
        j := j-1;
     fmentre
     v[j+1] := elem;
  fper
facció
```

Ordenació

(I) Ordenació per inserció

Anàlisi de costos:

- En el **millor cas**, el vector ja està ordenat: { 1, 2, 3, 4, 5 }
- L'algorisme ha de recórrer una vegada tota la seqüència per adonar-se'n que cada element està al seu lloc.

Ordenació

(I) Ordenació per inserció

Anàlisi de costos:

- En el **millor cas**, el vector ja està ordenat: { 1, 2, 3, 4, 5 }
- L'algorisme ha de recórrer una vegada tota la seqüència per adonar-se'n que cada element està al seu lloc.

Best case = O(n)

Ordenació

(I) Ordenació per inserció

Anàlisi de costos:

- En el **millor cas**, el vector ja està ordenat: { 1, 2, 3, 4, 5 }
- L'algorisme ha de recórrer una vegada tota la seqüència per adonar-se'n que cada element està al seu lloc.

Best case = O(n)

- En el **pitjor cas**, el vector està inversament ordenat: { 5, 4, 3, 2, 1 }

- En el **pitjor cas**, el vector està inversament ordenat: { 5, 4, 3, 2, 1 }

Exemple	Posició de la clau	Nombre de comparacions	Nombre de moviments	Total
El 4	Posició 2	1	1	2
El 3	Posició 3	2	2	4
El 2	Posició 4	3	3	6
L'1	Posició n	n-1	n-1	2(n-1)

En el pitjor cas s'ha de fer tot això, per tant: $2(1) + 2(2) + 3(2) + \dots + 2(n-1) = 2\,(1+2+3+\dots+n) = 2n(n-1)/2$

Worst case = $O(n^2)$

Ordenació

(I) Ordenació per inserció

Anàlisi de costos:

- En el **millor cas**, el vector ja està ordenat: { 1, 2, 3, 4, 5 }
- L'algorisme ha de recórrer una vegada tota la seqüència per adonar-se'n que cada element està al seu lloc.

Best case = O(n)

- En el **pitjor cas**, el vector està inversament ordenat: { 5, 4, 3, 2, 1 }

Worst case = $O(n^2)$

Ordenació

(I) Ordenació per inserció

Anàlisi de costos:

- En el **millor cas**, el vector ja està ordenat: { 1, 2, 3, 4, 5 }
- L'algorisme ha de recórrer una vegada tota la seqüència per adonar-se'n que cada element està al seu lloc.

Best case = O(n)

- En el **pitjor cas**, el vector està inversament ordenat: { 5, 4, 3, 2, 1 }

Worst case = $O(n^2)$

- En el **cas mig:**

Average case = $O(n^2)$