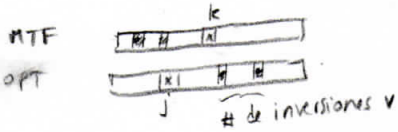
ALGORITMOS ONLINE.

Skies $ALG(n)\begin{cases} a\cdot n & \text{si } n\le k \\ a\cdot k+b & \text{si } n>k \end{cases}$ Si $k=\frac{b}{a}$ $\forall n\ge 1$

$\frac{ALG(n)}{OPT(n)} \le 2\left(2-\frac{a}{b}\right)$

Biblioteca
- $MTF(n) \le 2\cdot OPT(n)$ $\forall n\ge 1$
- $OPT(n)$ es $\Theta(n+p)$
- Trasponer es $\Theta(n\cdot p)$ no competitivo
- Conteo de frecuencia...



costo amortizado
$k + (k-1-v) - v$
(# de inv creadas, # inv solucionadas)
$= 2(k-v) - 1$
$\le 2j - 1$ $(k-v-1 \le j-1)$

Paginamiento en disco

FIFO, FC frecuency count, LRU least recently used, FWF flush when full

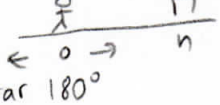
OPT = LFD

$\binom{n}{k} = \frac{n!}{(n-k)!\,k!}$

Stirling $\Theta(\ln n!) = \Theta(n\log n)$

$\sum_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r}$, $\sum_{i=0}^{n} 2^i = 2^{n+1}-1$

Estudiante borracho



Doblar distancia → girar 180°
es 9-competitivo

$2\left(\sum_{i=1}^{\log n+1} 2^i\right) + n$
$\le 2\cdot 2^{\log n+2} + n$
$= 2\cdot n\cdot 2^2 + n$
$= 9n$

Subasta: doubling. Vendedor se queda con todo $

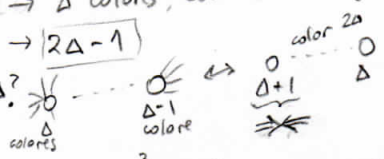
$\frac{ALG}{OPT} = \frac{\sum_{i=0}^{n+1} 2^i}{2^n+1} = \frac{2^{n+2}-1}{2^n+1} \le \frac{2^{n+2}}{2^n} = 4$

precio original

Colorear arcos OPT → Δ colores, con Δ = máx grado de nodo

∀ ALG → $2\Delta-1$

cota superior → 2Δ? paso $2\Delta-1$



cota inferior → caso part. Δ=4

necesita 7 colores

k server problem. k+1 puntos, A nuestro ALG

k adversarios óptimos. Para cada punto cubierto por A, hay un único Adv j que no tiene un servidor ahí.

Pedimos k+1. ALG mueve un serv. desde j a k+1
Adv j mueve un serv desde k+1 a j
$\sum Adv_i$ crece en 1, igual que A.

$A(\sigma) = \sum_{i=1}^{k} Adv_i(\sigma) \ge OPT(\sigma)$

$\Rightarrow A(\sigma) \ge k\,OPT(\sigma)$

Fork y vacas

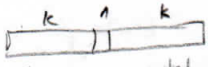
	b	d	
LAZYCOW	14.5	14.5	→ usa ALG 14.5 comp
SPLITUP	9	18	→ 1 vaca usa ALG 9-c

Feria Infinita: $k total

a) Sin devolución → Adversario vende 1 1 1 1 1 ① k k k k
ALG es k veces peor. hasta que en prod j el hombre compra. Luego vende todo a k después.

b) Con devolución → estrategia opt, comprar hasta gastar $\frac{k}{2}$, luego si encuentro prod. $s > \frac{k}{2}$, vendo todo y compro ese. Me aseguro gastar siempre $\$\frac{k}{2}$.

Tom y Jerry



Tom parte al medio

X• ir al plato más cerca del suelo no funca, pues agarro 1 y luego jerry me caga alternando

X• ir al que puedo alcanzar. Si no, no me muevo. Agarro 1 a un extremo y de ahí perdí ∞ al otro extremo

• Óptimo: tratar de quedarse a centro. 1 a un lado y el resto otro, salvo 1 de c/ 2k

ALGORITMOS PARALELOS

PRAM parallel random access memory

$T(n,p)$ = número de pasos en entrada de tamaño n con p procesadores, $p \ge 1$

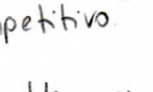
$S(n,p) = \frac{T(n,1)}{T(n,p)}$ → mejor alg secuencial conocido — speedup.

cuánto mejor es tener p proc. vs 1

$E(n,p) = \frac{S(n,p)}{p} = \frac{T(n,1)}{p\cdot T(n,p)}$, Eficiencia

cuánta mejora otorga cada procesador extra.

obtener máximo con torneo → paralelizable



$T(n, n/2) = \Theta(\log n)$
$S(n, n/2) = \Theta(n/\log n)$
$E(n, n/2) = \Theta(1/\log n)$

Para acercar E a 1 → $p = \Theta(n/\log n)$ (pues S es $n/\log n$)

ALG → tomar $\log n$ elts por c/ proc y calcular máx secuencialm
Luego empezar torneo

Lemma de Brent: Si existe algún alg CREW con $T(n,p) = t$ tal que $w(n) = s$ (trabajo total)

entonces existe alg CREW con $T(n, s/t) = \Theta(t)$

Ej máx: tenemos $T(n, n/2) = \Theta(\log n)$, $w(n) = \Theta(n)$ (cuanto cuesta impl torneo total)
$\Rightarrow \exists$ ALG $T(n, \Theta(n/\log n)) = \Theta(\log n)$

- AND con CRCW es $\Theta(1)$. OR También.
  con EREW es $\Theta(\log n)$

Merge paralelizable: a, b listas de entrada ordenadas, x lista de salida

merge(a, b, x, n) $|a| = |b| = n$

```
if n=1:
    x1 ← a1
    x2 ← b1
    compex(x, 1, 2) // compare + exchange
else:
    ao ← odds(a) ; bo ← odds(b)
    ae ← evens(a), be ← evens(b)
    merge(ao, bo, 0, n/2)
    merge(ae, be, e, n/2)
    shuffle(0, e, x)
    for i = 2: 2: 2n-2
        compex(x, i, i+1)
```

Pointer Jumping

Paginamiento en disco

FIFO, FC frecuency count, LRU least recently used, FWF flush when full

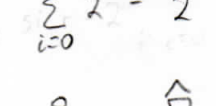
OPT = LFD

$\binom{n}{k} = \frac{n!}{(n-k)!\,k!}$

Stirling $\theta(\ln n!) = \theta(n \log n)$

$\sum_{i=0}^{n} r^i = \frac{1-r^{n+1}}{1-r}$; $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

Estudiante borracho



Doblar distancia → girar 180°

es 9-competitivo

$2 \cdot \left(\sum_{i=1}^{\log n+1} 2^i\right) + n$

$\leq 2 \cdot 2^{\log n + 2} + n$

$= 2 \cdot n \cdot 2^2 + n$

$= 9n$

Subasta: doubling. Vendedor se queda con todo $

$\frac{ALG}{OPT} = \frac{\sum_{i=0}^{n+1} 2^i}{2^n + 1} = \frac{2^{n+2} - 1}{2^n + 1} \leq \frac{2^{n+2}}{2^n} = 4$

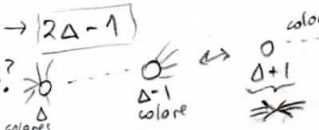
precio original

Colorear arcos

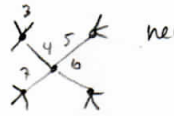
OPT → Δ colores, con Δ = máx grado de nodo

∀ ALG → $2\Delta - 1$

cota superior → 2Δ? paso $2\Delta - 1$



cota inferior → caso part. necesita 7 colores



k server problem. k+1 puntos, A nuestro ALG

k adversarios óptimos. Para cada punto cubierto por A, hay un único Advj que no tiene un servidor ahí.

Pedimos k+1. ALG mueve un serv. desde j a k+1

Advj mueve un serv desde k+1 a j

ΣAdvi crece en 1, igual que A.

$A(\sigma) = \sum_{i=1}^{k} Adv_i(\sigma) \geq OPT(\sigma)$

$\Rightarrow A(\sigma) \geq k\,OPT(\sigma)$

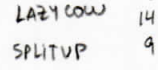
Fork y vacas

	b	d	
LAZY COW	14.5	14.5	→ usa ALG 14.5 comp
SPLITUP	9	18	→ 1 vaca usa ALG 9-c / la que encuentra ★

ALG 10-comp para dist

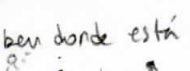
- SPLITUP con la vaca que se va por un rayo a $\frac{1}{9}$ de velocidad

VARIANTE PROBLEMA.



se ponen de acuerdo antes. OPT = $\frac{d}{2}$

a) saben donde está el este



cada vaca hace ALG 9-comp hasta llegar a $\frac{d}{2}$ → cada una demora $\frac{9d}{2}$-comp. → sumadas = 9-comp.

b) no saben dónde está el este.

Una se queda quieta, y la otra usa ALG 9-comp para llegar a la otra → recorre distancia d con ALG 9-comp.

$\frac{ALG}{OPT} \frac{9d}{\frac{d}{2}} \Rightarrow$ 18-comp.

a un extremo y • Óptimo: tratar de quedarse a centro. | otro, salvo 1 de c/ 2k

ALGORITMOS PARALELOS

PRAM parallel random access memory

T(n,p) = número de pasos en entrada de tamaño n con p procesadores, p>1

$S(n,p) = \frac{T(n,1)}{T(n,p)}$ → mejor alg. secuencial conocido — speedup.

cuánto mejor es tener p proc. vs 1

$E(n,p) = \frac{S(n,p)}{p} = \frac{T(n,1)}{p \cdot T(n,p)}$, Eficiencia

cuánta mejora otorga cada procesador extra.

obtener máximo con torneo → paralelizable

$T(n, n/2) = \theta(\log n)$

$S(n, n/2) = \theta(n/\log n)$

$E(n, n/2) = \theta(1/\log n)$

Para acercar E a 1 → $p = \theta(n/\log n)$ (pues S es $n/\log n$)

ALG → tomar log n elts por c/ proc y calcular máx secuencialm

Luego empezar torneo

Lemma de Brent: Si existe algún alg CREW con $T(n,p) =$ tal que $w(n) = S$ (trabajo total)

entonces existe alg CREW con $T(n, s/t) = \theta(t)$

Ej MÁX: tenemos $T(n, n/2) = \theta(\log n)$, $w(n) = \theta(n)$ (cuanto cuesta impl torneo total)

$\Rightarrow \exists$ ALG $T(n, \theta(n/\log n)) = \theta(\log n)$

- AND con CRCW es θ(1). OR También.
  con EREW es θ(log n)

Merge paralelizable:

a, b listas de entrada ordenadas, x lista de salida

merge(a, b, x, n) |a| = |b| = n

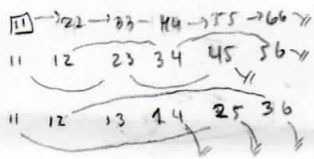
```
if n=1:
    x1 ← a1
    x2 ← b1
    compex(x, 1, 2) // compare + exchange
else:
    ao ← odds(a) ; bo ← odds(b)
    ae ← evens(a), be ← evens(b)
    merge(ao, bo, O, n/2)
    merge(ae, be, E, n/2)
    shuffle(O, E, X)
    for i = 2: 2: 2n-2
        compex(x, i, i+1)
```

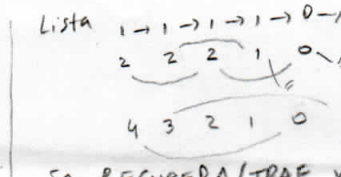
Pointer Jumping

- Reunir en listas enlazadas
- Prefix
- Euler Tour

Prefix compute

```
while ∃ i tq next[i] ≠ NULL:
    for i in parallel
        if next[i] ≠ Null
            y[next[i]] ← y[i] ⊗ y[next[i]]   (aquí esta la magi
        next[i] ← next[next[i]]
```





Multiplicar matrices paralelo.

ALG $\Theta(n)$ con n^2 procesadores

$C = A \cdot B \Rightarrow c_{ij} = \sum_k a_{ik} b_{kj}$

```
Prod(A,B,C)
    for procesador i,j
        c_ij ← ∅
        for k = 1 ... n
            c_ij ← c_ij + A_ik · B_kj
```

$p = n^2$

$T(n,p) = n$

$T(n,1) = n^3$

$w(n) = n^3$

$E(n) = \frac{n^3}{n^2 \cdot n} = 1$

MODELO CREW

generar n copias con n proc en $\Theta(\log n)$

Copy (x, B) solo P_1 conoce x. B es salida, mem comp.

```
Proc 1
    B[0] ← x
S ← 1
while S < n
    Proc j ; j ∈ [0, min(S, n-S)]
        B[j+S] ← B[j]
    S ← 2S
```

S representa cuantas copias de x están listas para ser copiadas.

$T(n,n) \in \Theta(\log n)$ pues se va duplicando.

¿$\Theta(1)$? → Brent. (con $w(n) = n$)

¿Las sumas pueden hacerse en tiempo $\Theta(\log n)$ con EREW (?)