



Sleeping
snail

Sort | Selection sort

Da es mehrere Voreinstellung in unserer Causation
während des Laufens.

sort($A[1..n]$: array of numbers): void

 for $i \leftarrow 1$ to $n-1$ do $\xrightarrow{\text{loop 1}}$

- 1) $\min \leftarrow i$
- 2) for $j \leftarrow i+1$ to n do $\xleftarrow{\text{loop 2}}$
 $i : A[i] < A[\min]$ then
 $\min \leftarrow j$
 $A[i \dots n]$
- 3) swap($A[i], A[\min]$)

UML ②: Wurde schon gesagt was der Punkt
über dem Klammerstrich \exists ist
um zu bestimmen
 $A[\min] = \min A[i \dots (j-1)]$ u

$$\min \in \{i \dots j-1\} \quad j \in \{i+1, \dots, n+1\}$$

Def: Nu scons occorre reç pag 1 u
no -toero u potepkox (i) $i \leq n-1$, to
unore, re:

$A[1 \dots (i-1)]$ e corpupor u
correspox(i-1)re kau -moxen emento

re sequa $A[1 \dots n]$ i $\in \{1, \dots, n\}$
Unoregux no i (unapusmo) $i=1$

Boza: $i = 1 \dots A[1 \dots (i-1)] \stackrel{d}{=} A[1 \dots 0] = A[\]$ ✓

Noapexak: kewa sda kxkoe unenherux u
kau -moxen emento re e vorezgo qz pag 5
curno unapusmo:

$A[1 \dots (i-1)]$ e corpupor u (e.g. 1-re
kau -moxen emento. ke vacuas $A[\dots]$)

Umwandlung einer LIFO-Liste in eine FIFO-Liste
 durch die Anwendung von $\text{push}()$ und $\text{pop}()$
 $\text{? } \text{list} \leq \text{list} \text{?}$. Umsetzung der Zeige-Punkte
 "verschieben" i.e.
 $A[1 \dots (\text{new}-1)]$ e copy from list in list
 $\bullet (\text{new}-1)$ the last element is $A[\text{t} \dots n]$.

$\text{? } \text{list} 2) \text{ min} = l$
~~in list min value is i~~
~~copy to list at $\text{list}[l]$~~
 ~~$\text{list}[l] = i$~~

Box 2:
 $\min = l$
 $A[\min] = A[l] = \min A[1 \dots (\text{new}-1)]$
 $\min = i \in 2, \text{ list}$

Voraussetzung: Mindestens 2 Werte vorhanden in list
 min \neq max
 $A[\min] = \min A[1 \dots (\text{new}-1)]$

Um den nächsten Knoten weiter zu gehen ist der Knoten mit dem niedrigsten Wert zu wählen. Dieser ist wiederum derjenige, der die geringste Distanz hat. Um diese zu bestimmen, müssen wir alle Nachbarn des aktuellen Knotens überprüfen.

Die Distanz eines Nachbarn ist die Summe der Distanz des aktuellen Knotens und der Abstand zwischen den beiden Knoten. Der Abstand zwischen zwei Knoten ist die Anzahl der Kanten, die auf dem kürzesten Pfad zwischen ihnen liegen. Wenn es mehrere Wege gibt, dann ist die Distanz des kürzesten Pfades.

Um die Distanz eines Nachbarn zu berechnen, müssen wir die Distanz des aktuellen Knotens mit der Distanz des Nachbarn vergleichen. Wenn der Nachbar einen niedrigeren Wert als der aktuelle Knoten hat, dann wird der aktuelle Knoten als Vorgänger des Nachbarn markiert und die Distanz des Nachbarn wird aktualisiert. Dieser Vorgang wiederholt sich, bis alle Knoten besucht sind.

Um die Distanz eines Nachbarn zu berechnen, müssen wir die Distanz des aktuellen Knotens mit der Distanz des Nachbarn vergleichen. Wenn der Nachbar einen niedrigeren Wert als der aktuelle Knoten hat, dann wird der aktuelle Knoten als Vorgänger des Nachbarn markiert und die Distanz des Nachbarn wird aktualisiert. Dieser Vorgang wiederholt sich, bis alle Knoten besucht sind.

Observe $\min_{\text{new}} = j$.

Comments:

$$A[\min_{\text{new}}] = \min A[i \dots (j_{\text{new}} - 1)].$$

$$\text{Cn.(2)} \quad A[j] \geq A[\min_{\text{new}}]$$

$$\begin{aligned} \text{Or } \bigcup_{i=0}^j \min_{\text{new}, i} &= \min A[i \dots (j_{\text{new}} - 1)] = \min A[i \dots j] \quad j = j_{\text{new}} - 1 \\ A[\min_{\text{new}}] &= \min A[i \dots (j_{\text{new}} - 1)] \\ &= \min A[i \dots (j_{\text{new}} - 1)] \quad j = j_{\text{new}} - 1 \\ &\cup B \text{ is present} \quad \min_{\text{new}} \in \{i \dots j_{\text{new}} - 1\} \\ &\min_{\text{new}} \in \{i \dots j_{\text{new}} - 1\} \quad \min A[i \dots (j_{\text{new}} - 1)] \end{aligned}$$

$$\{i, \dots, j-1\} \subsetneq \{i, \dots, j_{\text{new}} - 1\}$$

Implementation: T.e. $\text{new} \leftarrow i = n+1$
 Trace of unpopulated queue, see
 $A[\text{min}] = \min A[i \dots (i-1)] =$
 $= \min A[i \dots n]$ $(*)$
 $\min \in \{i \dots n\}$

Implementation C code is unq. ②
 Because of ~~the~~ ~~has~~ populating ~~to~~ unq. ①.
~~swap(A[i], A[min])~~ ~~has pop ⑥~~
~~u.n.~~ ~~$A[1 \dots (i-1)]$ e copiando in $C[i \dots n]$ (i-1)te h.s. en
 to $\min A[1 \dots n]$~~
 Or ~~(*)~~ ~~unq. see $A[\text{min}] = \min A[1 \dots n]$ u~~
~~n pu usn. ha pop. ⑥) $A[i] = \min A[1 \dots n]$~~
~~Tarea copiar $i_{\text{new}} = i+1 \leftrightarrow i = \underline{i_{\text{new}} - 1}$:~~

1) Or u.n. unsorted, re, se

$A[1 \dots (i-1)]$ e csg. $(i-1) - \text{te u.n. en. no}$ $\left\{ \text{***} \right\}$
mostra $A[1 \dots n]$ T.p. $A[i:j] = \min A[i \dots n]$
usc u nove, se $A[j:i] : A[j:j] \leq A[i:j]$

i.e. $A[1 \dots i]$ csg. L-te u.n. en. no
mostra $A[1 \dots i]$
unsorted $i = \text{new_}i$ T.p.

$A[1 \dots (\text{new_}i)]$ csg. $(\text{new_}i) - \text{te u.n. en. no}$
mostra $A[1 \dots n]$

2) Or u.n. unsorted $A[1 \dots (i-1)]$ e copiada.
Or word u.n. possever. se $C[\star]$ pos 3xarray,
se $A[1 \dots i]$ e copiada u csg $i = \text{new_}i$ \rightarrow
 $A[1 \dots (\text{new_}i)]$ e copiada.

Teorema lui L. Tărnăvanu: $\exists x \ i = n \quad i \notin A$

Ceră sături rezistență unei rea.

$A[1 \dots (i-1)]$ e copiul pos \cup
copiul $(i-1)$ -te h.s. er. nu secură $A[1 \dots n]$.

$A[1 \dots (n-1)]$ e copiul pos \cup
copiul $(n-1)$ -te h.s. er. nu secură $A[1 \dots n]$

+

T.c. $A[n]$ e h.s. er. nu secură $A[1 \dots n]$
 $A[1 \dots n]$ e copiul pos \rightarrow

Kann ich diese Σ -Berechnung umsetzen?

$$- n - 1$$

Konventionelleweise ist bestimmt, wenn man Σ rechts schreibt.

$$\begin{aligned} \sum_{i=0}^b 1 &= b-a+1 \\ \sum_{i=1}^{n-1} 1 - \sum_{i=1}^{n-1} 1 &= \\ \sum_{i=1}^{n-1} (-1) &= \\ \sum_{i=1}^{n-1} (n-i) &= \\ = n \cdot (n-1) &= \frac{(n-1) \cdot n}{2} = \frac{n^2 - n}{2} \end{aligned}$$

302 / Kodane: rekurzivní funkce na maxima

Kodane ($A[1..n]$: sestava z n čísel): Number

- 1) localmax $\leftarrow 0$
- 2) globalmax $\leftarrow -\infty$
- 3) for $i \leftarrow 1$ to n
 do
 line

- 4) localmax $\leftarrow \max(A[i], \underline{\text{localmax}}, \underline{\text{globalmax}})$
- 5) globalmax $\leftarrow \max(\underline{\text{localmax}}, \underline{\text{globalmax}}, A[i])$
- 6) return globalmax

$a > b \rightarrow a \max(a, b) = a$
 $a < b \rightarrow a \max(a, b) = b$

$\left. \begin{array}{l} \text{localmax - hotí - rekurzivní funkce} \\ \text{globalmax - hotí - rekursivní funkce} \end{array} \right\}$ funkce
 $i \in \{1, \dots, n+1\}$ $\left. \begin{array}{l} \text{globalmax} \leftarrow A[1], \text{globalmax} \leftarrow A[2], \dots, \text{globalmax} \leftarrow A[i-1] \end{array} \right\}$

Uniqueness no i:

$$\text{Base: } \underbrace{\text{local max} = 0}_{i=1}, \underbrace{\text{global max} = -\infty}_{i=1 \dots (i-1)} = A[1 \dots 0] = ATJ \quad (\vee)$$

Recurrence: ~~Se hævor vi en, kendte lokale maks og
og en ikkeupdateret. Urban. Denne lokale maks
og næste lokale maks $i_{new} = i+1$ vil være inden
værdien af vore i og derfor dog nu
være mindre.~~

e.B. hvis $i > x$:

$\text{localmax}_i = \text{localmax}_{i-1}$ = max sum of subarray finishing with $A[i-1]$

$\text{globalmax}_i = \text{max sum of subarray finishing
with } ATJ \text{ or } A[2 \dots i] \text{ or } \dots, A[i-1]$

Cn.1 $\underbrace{\text{localmax} + A[i]}_{\text{global max} < \text{local max}} \geq ATJ$

Cn.2 $\underbrace{\text{localmax} + A[i]}_{\text{local max} > \text{global max}} > ATJ \text{ & global max} \geq \text{local max}$

(C1.3) localmax + A[i] ≤ A[i] & globalmax < localmax

(C1.4) localmax + A[i] ≤ A[i] & localmax ≤ globalmax

(C1.1) localmax + A[i] ≥ A[i]
globalmax > localmax_{new}

- (localmax_{new} = localmax + A[i])
globalmax_{new} = max{localmax + A[i]}

(C1.2) localmax = max sum of subarray with last element A[i-1]

(C1.3) localmax_{new} = max sum of subarray with last element A[i-1] = A[[inew-1]]

$$i = i_{\text{new}} - 1$$

(Or u.n globalmax = max sum of subarray with last element A[1:i-1] = ... or A[i-1])

(*) + (*) non-recursive globalmax_{new} = max sum of subarray with last element A[i] or A[2:i] or ... or [i_{\text{new}}-1]

Terminator: $i = n+1$

Or we have to print max ce cdo.

Let's run $A[1:n+1]$ - $n+1$ loops
to print max value in array $A[1:n+1]$

Persons to global max ce $c[i..n]$.
maximize or $A[i..n]$

$i = n+1$
 $\rightarrow 1$

$A[1:n+1]$

Q2 (A[1...n] : array of numbers) : set

- 1) ~~i = 1~~
- 2) while $A[i] \leq 0$ and $i \leq n$ do
 3) $i \leftarrow i + 1$
- 4) if $i = n + 1$
 5) return EmptySet
- 6) i $\leftarrow i$
- 7) temp $\leftarrow A[i]$
- 8) max $\leftarrow 0$, maxi $\leftarrow 0$, maxj $\leftarrow 0$
- 9) while $i \leq n$
 10) if temp $> \max$
 11) max $\leftarrow \text{temp}$
 12) maxi $\leftarrow i$
 13) maxj $\leftarrow j$
- 14) i $\leftarrow i + 1$
- 15) temp $\leftarrow \text{temp} + A[i:j]$
- 16) if temp ≤ 0
 17) temp $\leftarrow 0$
- 18) i $\leftarrow i + 1$
- 19) return (\max_i , \max_j)

Приложение к рекуррентным уравнениям.

Несколько случаев решения рекуррентных уравнений

Рекуррентные со свойствами.

$$T(n) = \underbrace{8 \cdot T(n-1)}_{\text{стационарный член}} - \underbrace{15 \cdot T(n-2)}_{\text{линейный член}} + \underbrace{(8 \cdot n^1 + 5 \cdot n^0) \cdot 4^n}_{\text{вспомогательный член}} + \underbrace{n^0 3^n}_{\text{вспомогательный член}}$$

Рекуррентные с константой и с параметром.

$$x^n = 8 \cdot x^{n-1} - 15 \cdot x^{n-2} \quad // \quad x^{n-2} \neq 0$$

$$\begin{aligned} x^2 - 8 \cdot x + 15 &= 0 \\ D = 4 &\rightarrow x_{1,2} = \frac{8 \pm 2}{2} \end{aligned} \quad \left. \begin{array}{l} x_1 = 5 \\ x_2 = 3 \end{array} \right\} \rightarrow \left. \begin{array}{l} x_1 = 5 \\ x_2 = 3 \end{array} \right\} y_m$$

Случай рекуррентного уравнения вида:

(стационарный член и вспомогательный член) $\rightarrow \text{декр}(n) + 1$

$$f(n) = (8 \cdot n^3 + 5 \cdot n^2) \cdot 3^n + 2 \cdot \underbrace{n^2 \cdot 1^n}_{\text{oder } 1} + 2 \cdot 4 \cdot 4^n \cdot 3^n + 1 \cdot 1 \cdot 1 \cdot 1 \cdot 3^n$$

Beobachtung machen wir:

$$2 \cdot 3 \cdot 5 \cdot 3^n + 2 \cdot 4 \cdot 3^n + 1 \cdot 1 \cdot 1 \cdot 1 \cdot 3^n =$$

$$= 2 \cdot 4 \cdot 1^n + 2 \cdot 3 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 5 \cdot 3^n$$

$$\begin{aligned} T(n) &= (C_1 + C_2 \cdot n + C_3 \cdot n^2 + C_4 \cdot n^3) \cdot 1^n + \\ &\quad (C_5 + C_6 \cdot n) \cdot 3^n + \\ &\quad (C_7 + C_8 \cdot n) \cdot 4^n + \end{aligned}$$

$$\underbrace{C_3 \cdot 5^n}$$

$$\overbrace{T(n)} \approx 5^n$$

Wunderbarne

$$\sum_{i=1}^{n-k+1} \log_{\alpha} -k > -1$$
$$\log_{\alpha} -k = -1$$
$$-k < -1$$
$$k > -1$$
$$\sum_{i=1}^{\lceil \frac{n}{2} \rceil} -1 = b - \alpha + 1$$
$$\sum_{i=\lceil \frac{n}{2} \rceil + 1}^n -1 = b - \alpha + 1$$
$$\sum_{i=1}^{\lceil \frac{n}{2} \rceil} -1 = \sum_{i=1}^{\lceil \frac{n}{2} \rceil} -1$$
$$\sum_{i=1}^{\lceil \frac{n}{2} \rceil} -1 = \sum_{i=1}^{\lceil \frac{n}{2} \rceil} -1$$

*

*

$$T(n) = T(n-1) + n^{1/3}$$

$$\text{const} \sum_{i=1}^{\lceil n^{1/3} \rceil} i^{1/3} = n$$

$$\begin{aligned}
 T(n) &= T(n-1) + n^{1/3} \\
 &= T(n-3) + (n-2)^{1/3} + (n-1)^{1/3} + n^{1/3} = \dots \\
 &= T(0) + 1^{1/3} + 2^{1/3} + \dots + (n-(n-1))^{1/3} \\
 &\quad (n-(n-1))^{1/3} (n-(n-2))^{1/3} \\
 &= T(0) + \sum_{i=1}^{\lceil n^{1/3} \rceil} i^{1/3}
 \end{aligned}$$

$$T(n) = T(n-2) + (n-1)^{1/3} + n^{1/3} =$$

$$T(n-1) = T(n-2) + (n-1)^{1/3}$$

$$T(n) = T(n-1) + n^{1/3}$$

$$T(n) = T(n-1) + \sqrt[3]{n}$$

 $T(n) = 2 \cdot T(n-1) - T(n-2)$

$$x^n = 1 \cdot x^{n-1} - x^{n-2} //: x^{n-2} \neq 0$$

$$x^2 - 2 \cdot x^1 + 1 = 0$$

$$x_1 = x_2 = 1$$

$$\{1\} \text{ in}$$

$$\text{Danklert \& Ong: } T(n) = (c_1 \cdot n^0 + c_2 \cdot n^1) \cdot 1^n \underset{n}{\approx} n$$

$$c_1 + c_2 \cdot n \quad c_2 > 0$$

 $T(n) = \Theta(n)$

$$T(n) = \Theta(n)$$

constant

$$\begin{aligned} &= T(0) + 4 \cdot \sum_{i=1}^{\lfloor n/4 \rfloor} i \cdot \underbrace{T\left(\frac{n}{4} - 1\right)}_{\text{constant}} \\ &= T(0) + 4 \cdot \sum_{i=1}^{\lfloor n/4 \rfloor} i \cdot \underbrace{T\left(\frac{n}{4} - 1\right)}_{\text{constant}} \end{aligned}$$

$$T(n) = T(n-2) + 4 \cdot (n-1) - 10 = \dots =$$

$$T(n-1) = T(n-2) + 4 \cdot (n-1) - 10$$

$$T(n) = T(n-1) + 4 \cdot \underbrace{n}_{\text{constant}}$$

$$T(n) = T(n-1) + \frac{4n}{4}$$

base case

300 5

$$T(n) = T(n-1) + 1$$

[H(w)]

represents power bank

~~300~~ 60

$$\tau(n) = \tau(n-1) + 1/n$$

$H(\omega)$

repet passus bone

$$\text{H}(n) = n \cdot T(n-1)$$

$\boxed{H(n)}$ represents recursive rule

$$\text{Bsp: } T(n) = 2 \cdot T(n-1) + \frac{1}{n}$$

$$T(n) = \underbrace{2 \cdot T(n-1)}_{T(n-1)} + n^{-1}$$

$$T(n-1) = 2 \cdot T(n-2) + (n-1)^{-1} \\ \Rightarrow T(n) = 2 \cdot (2 \cdot T(n-2) + (n-1)^{-1}) + n^{-1} =$$

$$T(n-2) = \underbrace{2 \cdot T(n-3)}_{T(n-3)} + \frac{1}{(n-2)^{-1}}$$

$$= \underline{\underline{2^2 \cdot T(n-2) + 2 \cdot (n-1)^{-1} + n^{-1}}} =$$

$$= 2^2 \cdot (2 \cdot T(n-3) + (n-2)^{-1}) + 2 \cdot (n-1)^{-1} + n^{-1} =$$

$$= 2^2 \cdot T(n-3) + \underbrace{2^2 \cdot (n-2)^{-1}}_{2^1 \cdot (n-2)^{-1}} + \underbrace{2^1 \cdot (n-1)^{-1} + n^{-1}}_{\dots} = \dots$$

$$= 2^n \cdot T(0) + 2^{n-1} \cdot 1^{-1} + 2^{n-2} \cdot 2^{-1} + \dots + \underline{\underline{n^{-1}}} =$$

$$= 2^n \cdot T(0) + \sum_{i=0}^{n-1} 2^i \cdot (n-i)^{-1} = \text{ouge vorzunehmen}$$

$$= T(O) \cdot 2^n + \sum_{i=1}^n \frac{2^{n-i}}{i} = (\text{no-yogaden } \text{zu } O \text{ ergibt})$$

$$= T(O) \cdot 2^n + 2^n \cdot \sum_{i=1}^n \frac{2^{-i}}{i} =$$

$$= \boxed{T(O) \cdot 2^n + 2^n \cdot \sum_{i=1}^n \frac{1}{2^i \cdot i}} \quad (*)$$

die rechte Seite
wir spezifizieren

$$\text{Cesq} \quad \sum_{i=1}^n \frac{1}{2^i \cdot i} \leq \sum_{i=1}^n \frac{1}{2^i} = \sum_{i=1}^n \left(\frac{1}{2}\right)^i \leq \sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i = \frac{1}{2} = 1$$

$$\text{T.e.} \quad \sum_{i=1}^n \frac{1}{2^i \cdot i} \leq 1 \quad \text{take } 2^n \cdot \sum_{i=1}^n \frac{1}{2^i \cdot i} \leq 2^n$$

T.e.
 $(*) \leq 2^n$ und $T(n) \asymp 2^n$

$$v + (1-v) \cdot \frac{1+n}{n} = (n) \cdot \boxed{\text{H}(u)}$$

reduced procedure

task f (n: unsigned int): int
S<0

while S <= n
S ← 5 · S + 3

return S

- a) Berechnen Sie die Laufzeit f(n):
b) Berechnen Sie den Wert T(0):

3 a) 8n + 18
b)

$$\text{Zeigt (1)} \quad T(n) = 4 \cdot T(\sqrt{n}) + n$$

ist ein Rekurrenz-Gleichung.

Задача 12

```
f(n: unsigned int) : int
if n=0
    return 0
```

```
    return 0
    n ← n * n + 1
    return n / 41
```

```
for k ← 1 to n - 1
    Q ← Q - f(k)
```

```
return Q
```

$f(n) = ?$ $\cup T(n) \approx ?$

Задача 12