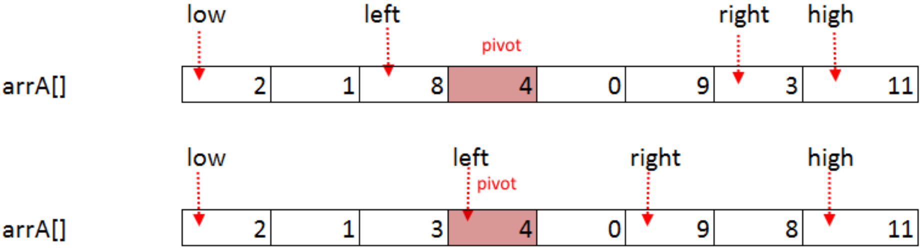
Algorithmen I - Tutorium 6

Sebastian Schmidt – *isibboi@gmail.com*

Arbeitsgruppe Kryptographie und Sicherheit



Quicksort (abstrakt)

Function quickSort(s : Sequence **of** Element) : Sequence **of** Element
  if $|s| \leq 1$ **then return** s
  pick "some" $p \in s$
  $a := \langle e \in s : e < p \rangle$
  $b := \langle e \in s : e = p \rangle$
  $c := \langle e \in s : e > p \rangle$
  return concatenation of quickSort(a), b, and quickSort(c)

Beispiel: 5, 1, 0, 2, 4, 3, 6
Pivotwahl: Rechts

Quicksort (abstrakt)

Function quickSort(s : Sequence **of** Element) : Sequence **of** Element
    if $|s| \leq 1$ **then return** s
    pick "some" $p \in s$
    $a := \langle e \in s : e < p \rangle$
    $b := \langle e \in s : e = p \rangle$
    $c := \langle e \in s : e > p \rangle$
    return concatenation of quickSort(a), b, and quickSort(c)

Was ist eine Worst-Case-Eingabe, wenn als Pivot immer das rechte Element gewählt wird?

Quicksort (abstrakt)

Function quickSort(s : Sequence **of** Element) : Sequence **of** Element
  if $|s| \leq 1$ **then return** s
  pick "some" $p \in s$
  $a := \langle e \in s : e < p \rangle$
  $b := \langle e \in s : e = p \rangle$
  $c := \langle e \in s : e > p \rangle$
  return concatenation of quickSort(a), b, and quickSort(c)

Was ist eine Best-Case-Eingabe mit sieben Elementen, wenn als Pivot immer das rechte Element gewählt wird?

Quicksort (abstrakt)

Function quickSort(s : Sequence **of** Element) : Sequence **of** Element
  if $|s| \leq 1$ **then return** s
  pick "some" $p \in s$
  $a := \langle e \in s : e < p \rangle$
  $b := \langle e \in s : e = p \rangle$
  $c := \langle e \in s : e > p \rangle$
  return concatenation of quickSort(a), b, and quickSort(c)

Wie muss man das Pivot wählen, um eine aufsteigend sortierte Folge schnell zu sortieren?

Quicksort (abstrakt)

Function quickSort(s : Sequence **of** Element) : Sequence **of** Element
  if $|s| \leq 1$ **then return** s
  pick "some" $p \in s$
  $a := \langle e \in s : e < p \rangle$
  $b := \langle e \in s : e = p \rangle$
  $c := \langle e \in s : e > p \rangle$
  return concatenation of quickSort(a), b, and quickSort(c)

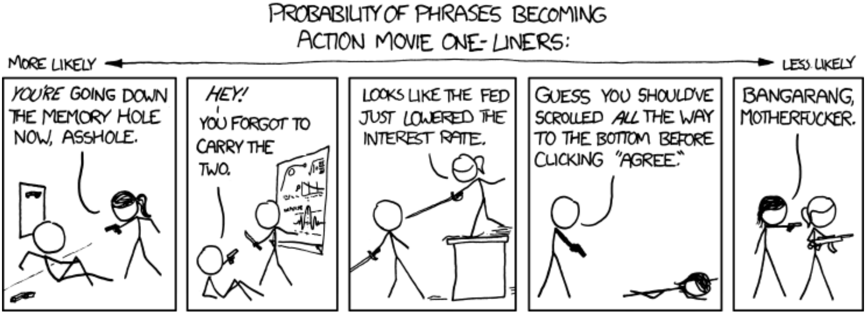
Wie wählt man das perfekte Pivot?
Ist das praktikabel?

Quicksort (abstrakt)

Function quickSort(s : Sequence **of** Element) : Sequence **of** Element
  if $|s| \leq 1$ **then return** s
  pick "some" $p \in s$
  $a := \langle e \in s : e < p \rangle$
  $b := \langle e \in s : e = p \rangle$
  $c := \langle e \in s : e > p \rangle$
  return concatenation of quickSort(a), b, and quickSort(c)

Welche Pivotwahl-Heuristik würdet ihr wählen, wenn ihr Quicksort implementieren müsstet?

Pause



PROBABILITY OF PHRASES BECOMING ACTION MOVIE ONE-LINERS:
MORE LIKELY
LESS LIKELY
YOU'RE GOING DOWN THE MEMORY HOLE NOW, ASSHOLE.
HEY! YOU FORGOT TO CARRY THE TWO.
LOOKS LIKE THE FED JUST LOWERED THE INTEREST RATE.
GUESS YOU SHOULD'VE SCROLLED ALL THE WAY TO THE BOTTOM BEFORE CLICKING "AGREE."
BANGARANG, MOTHERFUCKER.

Exkurs: `std::sort`

- Standardfunktion aus der C++-STL
- Der vermutlich meistbenutzte Sortieralgorithmus der Welt
- Funktionsweise:
  - Startet mit halbrekursivem median of three Quicksort
  - Wenn der Quicksort eine Partitionsgröße kleiner oder gleich 16 erreicht, wird auf Insertionsort gewechselt
  - Wenn die Rekursionstiefe vom Quicksort $2\log_2(n)$ übersteigt, wird für die entsprechende Partition auf Heapsort gewechselt
  - Der Heapsort wird ausgeführt, bis der Heap die Größe 16 hat, dann wird auf Insertionsort gewechselt

Warum startet man mit Quicksort?

Exkurs: `std::sort`

- Standardfunktion aus der C++-STL
- Der vermutlich meistbenutzte Sortieralgorithmus der Welt

- Funktionsweise:
  - Startet mit halbrekursivem median of three Quicksort
  - Wenn der Quicksort eine Partitionsgröße kleiner oder gleich 16 erreicht, wird auf Insertionsort gewechselt
  - Wenn die Rekursionstiefe vom Quicksort $2\log_2(n)$ übersteigt, wird für die entsprechende Partition auf Heapsort gewechselt
  - Der Heapsort wird ausgeführt, bis der Heap die Größe 16 hat, dann wird auf Insertionsort gewechselt

Warum startet man mit Quicksort?

Exkurs: `std::sort`

- Standardfunktion aus der C++-STL
- Der vermutlich meistbenutzte Sortieralgorithmus der Welt

- Funktionsweise:
  - Startet mit halbrekursivem median of three Quicksort
  - Wenn der Quicksort eine Partitionsgröße kleiner oder gleich 16 erreicht, wird auf Insertionsort gewechselt
  - Wenn die Rekursionstiefe vom Quicksort $2\log_2(n)$ übersteigt, wird für die entsprechende Partition auf Heapsort gewechselt
  - Der Heapsort wird ausgeführt, bis der Heap die Größe 16 hat, dann wird auf Insertionsort gewechselt

Warum startet man mit Quicksort?

Exkurs: `std::sort`

- Standardfunktion aus der C++-STL
- Der vermutlich meistbenutzte Sortieralgorithmus der Welt

- Funktionsweise:
  - Startet mit halbrekursivem median of three Quicksort
  - Wenn der Quicksort eine Partitionsgröße kleiner oder gleich 16 erreicht, wird auf Insertionsort gewechselt
  - Wenn die Rekursionstiefe vom Quicksort $2\log_2(n)$ übersteigt, wird für die entsprechende Partition auf Heapsort gewechselt
  - Der Heapsort wird ausgeführt, bis der Heap die Größe 16 hat, dann wird auf Insertionsort gewechselt

Warum benutzt man auf kleinen Partitionen Insertionsort?

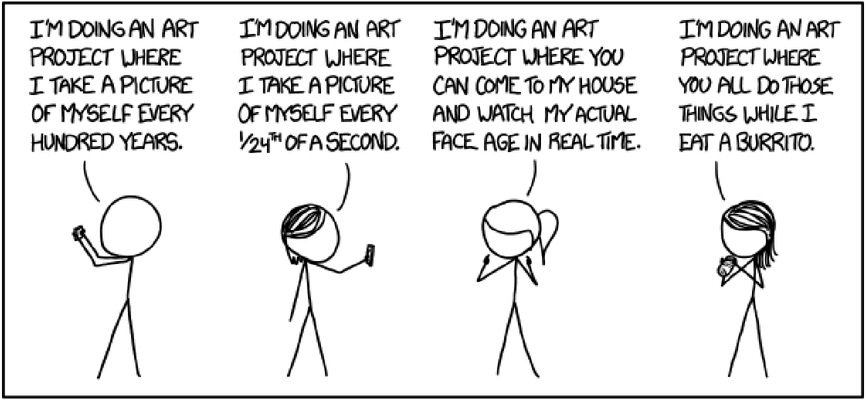
Exkurs: `std::sort`

- Standardfunktion aus der C++-STL
- Der vermutlich meistbenutzte Sortieralgorithmus der Welt

- Funktionsweise:
  - Startet mit halbrekursivem median of three Quicksort
  - Wenn der Quicksort eine Partitionsgröße kleiner oder gleich 16 erreicht, wird auf Insertionsort gewechselt
  - Wenn die Rekursionstiefe vom Quicksort $2\log_2(n)$ übersteigt, wird für die entsprechende Partition auf Heapsort gewechselt
  - Der Heapsort wird ausgeführt, bis der Heap die Größe 16 hat, dann wird auf Insertionsort gewechselt

Warum benutzt man Heapsort?

Pause



I'M DOING AN ART PROJECT WHERE I TAKE A PICTURE OF MYSELF EVERY HUNDRED YEARS.
I'M DOING AN ART PROJECT WHERE I TAKE A PICTURE OF MYSELF EVERY 1/24TH OF A SECOND.
I'M DOING AN ART PROJECT WHERE YOU CAN COME TO MY HOUSE AND WATCH MY ACTUAL FACE AGE IN REAL TIME.
I'M DOING AN ART PROJECT WHERE YOU ALL DO THOSE THINGS WHILE I EAT A BURRITO.

Quicksort (konkret)

Function partition(a : **Array of** Element; $\ell, r, k : \mathbb{N}$)

  $p := a[k]$

  swap$(a[k], a[r])$

  $i := \ell$

  for $j := \ell$ **to** $r-1$ **do**

    invariant $\begin{array}{l|c|c|c|c|}
 & \ell \quad & i \quad & j \quad & r \\
\hline
 & \leq p & > p & ? & p \\
\hline
\end{array}$

    if $a[j] \leq p$ **then**

      swap$(a[i], a[j])$

      $i{+}{+}$

  assert $\begin{array}{l|c|c|c|}
 & \ell \quad & i \quad & r \\
\hline
 & \leq p & > p & p \\
\hline
\end{array}$

  swap$(a[i], a[r])$

  assert $\begin{array}{l|c|c|c|}
 & \ell \quad & i & \quad r \\
\hline
 & \leq p & p & > p \\
\hline
\end{array}$

  return i

Beispiel:
3, 1, 5, 6, 4, 1, 3

Pivotwahl: Rechts

Quicksort (konkret)

Karlsruher Institut für Technologie

Function partition(a : **Array of** Element; $\ell, r, k : \mathbb{N}$)

$p := a[k]$

swap($a[k], a[r]$)

$i := \ell$

for $j := \ell$ **to** $r-1$ **do**

invariant $\begin{array}{l|c|c|c|c|} & \ell & i & j & r \\ \hline & \le p & > p & ? & p \\ \hline \end{array}$

if $a[j] \le p$ **then**

swap($a[i], a[j]$)

$i{+}{+}$

assert $\begin{array}{l|c|c|c|} & \ell & i & r \\ \hline & \le p & > p & p \\ \hline \end{array}$

swap($a[i], a[r]$)

assert $\begin{array}{l|c|c|c|} & \ell & i & r \\ \hline & \le p & p & > p \\ \hline \end{array}$

return i

Was passiert, wenn alle Elemente gleich sind?

Quicksort (konkret)

Function partition(a : **Array of** Element; $\ell, r, k : \mathbb{N}$)

  $p := a[k]$

  swap$(a[k], a[r])$

  $i := \ell$

  for $j := \ell$ **to** $r - 1$ **do**

    invariant $\begin{array}{cccc} \ell & i & j & r \\ \hline \multicolumn{1}{|c|}{\leq p} & \multicolumn{1}{c|}{> p} & \multicolumn{1}{c|}{?} & \multicolumn{1}{c|}{p} \\ \hline \end{array}$

    if $a[j] \leq p$ **then**

      swap$(a[i], a[j])$

      $i{+}{+}$

  assert $\begin{array}{ccc} \ell & i & r \\ \hline \multicolumn{1}{|c|}{\leq p} & \multicolumn{1}{c|}{> p} & \multicolumn{1}{c|}{p} \\ \hline \end{array}$

  swap$(a[i], a[r])$

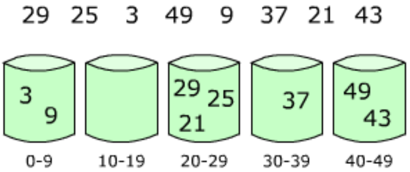
  assert $\begin{array}{ccc} \ell & i & r \\ \hline \multicolumn{1}{|c|}{\leq p} & \multicolumn{1}{c|}{p} & \multicolumn{1}{c|}{> p} \\ \hline \end{array}$

  return i

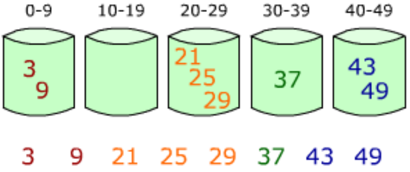
Was kann man machen, um diesen Worst-Case zu verhindern?

Bucketsort

1. Elemente ihren Buckets zuordnen:



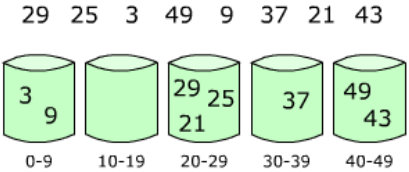
2. Innerhalb der Buckets sortieren:



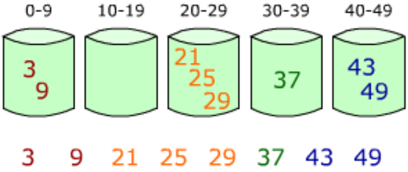
Wie sortiert man innerhalb der Buckets?

Bucketsort

1. Elemente ihren Buckets zuordnen:



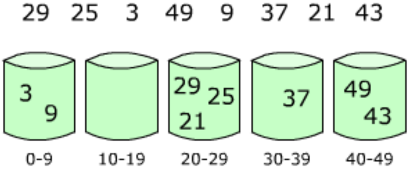
2. Innerhalb der Buckets sortieren:



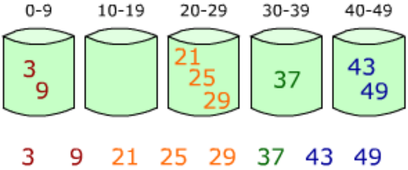
Angenommen, es gibt eine Funktion f, die jedem Element in $O(1)$ Zeit sein Bucket zuordnet. Wie ist die Best-Case-Laufzeit von Bucketsort?

Bucketsort

1. Elemente ihren Buckets zuordnen:



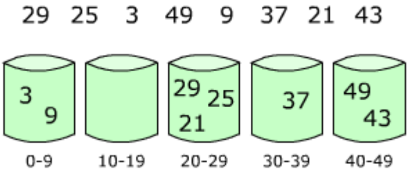
2. Innerhalb der Buckets sortieren:



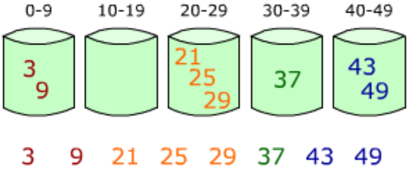
Angenommen, es gibt eine Funktion f, die jedem Element in $O(1)$ Zeit sein Bucket zuordnet. Wie ist die Worst-Case-Laufzeit von Bucketsort?

Bucketsort

1. Elemente ihren Buckets zuordnen:

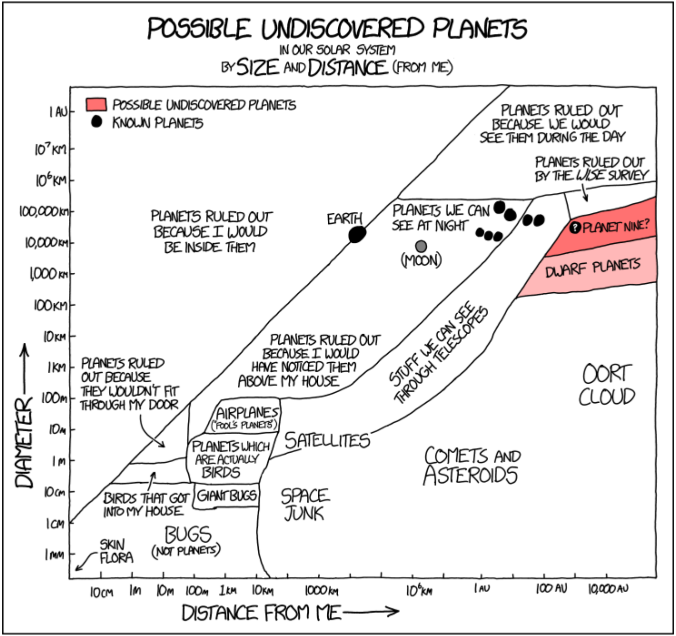


2. Innerhalb der Buckets sortieren:



Angenommen, es gibt eine Funktion f, die jedem Element in $O(1)$ Zeit sein Bucket zuordnet. Wie ist die Average-Case-Laufzeit von Bucketsort?

Ende


POSSIBLE UNDISCOVERED PLANETS
IN OUR SOLAR SYSTEM
BY SIZE AND DISTANCE (FROM ME)
POSSIBLE UNDISCOVERED PLANETS
KNOWN PLANETS
PLANETS RULED OUT BECAUSE WE WOULD SEE THEM DURING THE DAY
PLANETS RULED OUT BY THE WISE SURVEY
PLANETS RULED OUT BECAUSE I WOULD BE INSIDE THEM
EARTH
PLANETS WE CAN SEE AT NIGHT
(MOON)
PLANET NINE?
DWARF PLANETS
PLANETS RULED OUT BECAUSE I WOULD HAVE NOTICED THEM ABOVE MY HOUSE
STUFF WE CAN SEE THROUGH TELESCOPES
PLANETS RULED OUT BECAUSE THEY WOULDN'T FIT THROUGH MY DOOR
AIRPLANES ("FOOL'S PLANETS")
PLANETS WHICH ARE ACTUALLY BIRDS
SATELLITES
OORT CLOUD
COMETS AND ASTEROIDS
BIRDS THAT GOT INTO MY HOUSE
GIANT BUGS
SPACE JUNK
SKIN FLORA
BUGS (NOT PLANETS)
1 AU
10^7 KM
10^6 KM
100,000 KM
10,000 KM
1,000 KM
100 KM
10 KM
1 KM
100 M
10 M
1 M
10 CM
1 CM
1 MM
DIAMETER
10 CM
1 M
10 M
100 M
1 KM
10 KM
1000 KM
10^6 KM
1 AU
100 AU
10,000 AU
DISTANCE FROM ME