Tölvunarfræði 2

Vika 7

Eiríkur Ernir Þorsteinsson

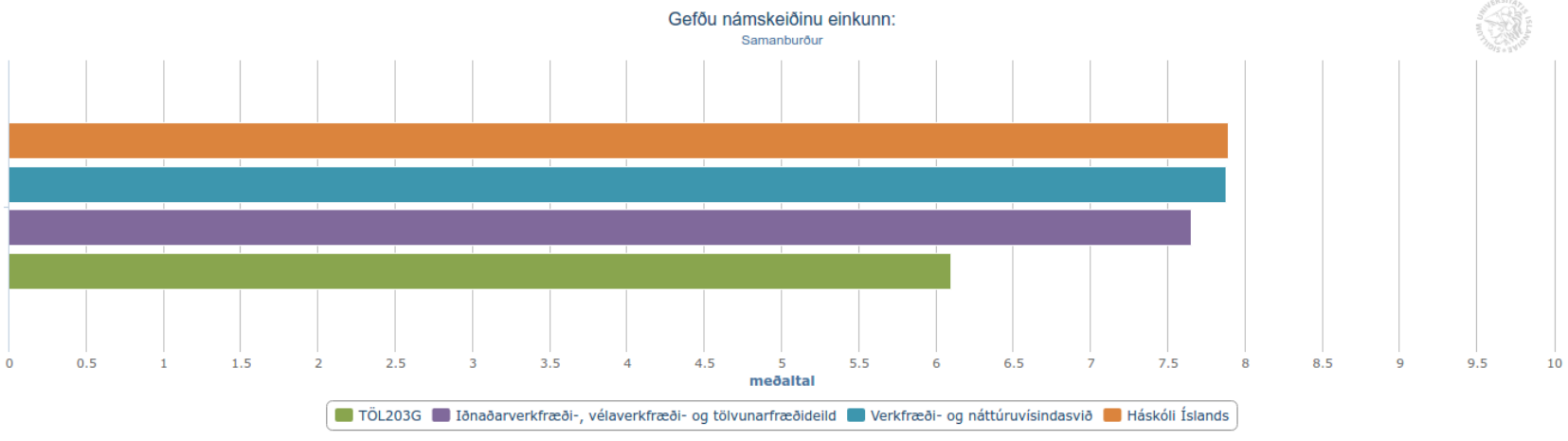
Háskóli Íslands

Vor 2017

HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRUVÍSINDASVIÐ

Yfirlit

Miðmisseriskönnun



Meðaleinkunn 6.1, 45% þátttaka

Atriði sem komu fram

- Gott:
  - Piazza

Atriði sem komu fram

- Gott:
  - Piazza
- Verra:
  - **HEIMADÆMI ALLT OF LÖNG OG ERFIÐ**
  - Salurinn er ekki nógu góður
  - Dæmatímar illa nýttir
  - Fyrirlestrar illa nýttir
- Listarnir eru ekki tæmandi, en þessi atriði voru nefnd endurtekið

Breytingar

- Heimadæmi verða kerfisbundið einfölduð
  - Tvö einföld, tvö eins og þau hafa verið
- Dæmatímakennarar sjá um að setja inn nemendalausnir
  - Uppfærðir skilmálar á heimadæmablöðum, mikilvægt að lesa!
- Meiri töflukennsla í dæmatímum
  - Mikilvægt að mæta undirbúin með spurningu!
- Við erum stödd í Háskólatorgi, HT-105
- Skoðanakönnun um framtíð yfirferðar:
  https://goo.gl/forms/2IPHR77I2DsfpzHl2

Yfirlit

Myndrænn samanburður

- Valröðun: Glæra 17 á Elementary sorts
- Innsetningarröðun: Glæra 25 í Elementary sorts
- Ofansækin sameiningarröðun: Glæra 9 í Merge sort
- Neðansækin sameiningarröðun: Glæra 24 í Merge sort

Selection sort inner loop

To maintain algorithm invariants:

- Move the pointer to the right.

```
i++;
```

- Identify index of minimum entry on right.

```
int min = i;
for (int j = i+1; j < N; j++)
   if (less(a[j], a[min]))
      min = j;
```

- Exchange into position.

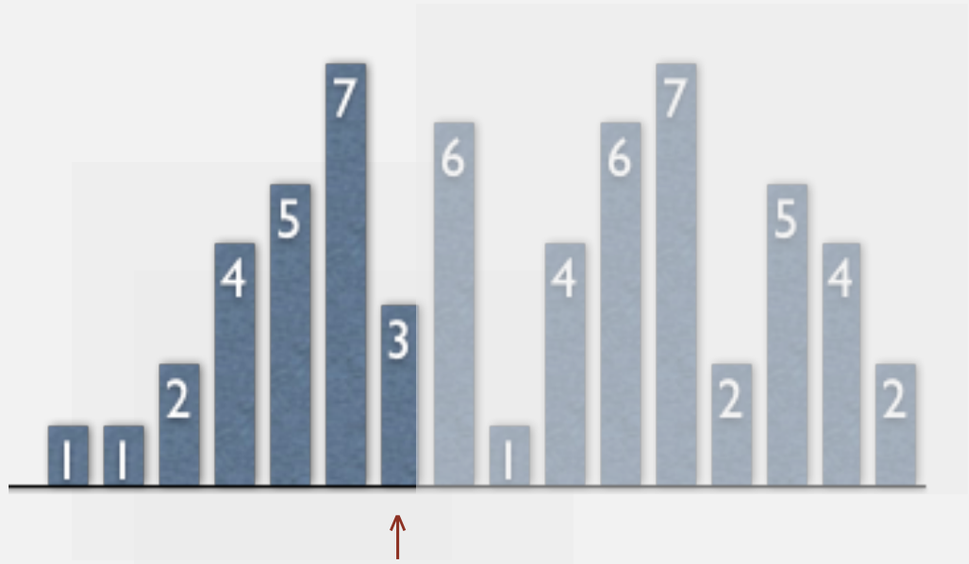
```
exch(a, i, min);
```

Insertion sort inner loop

To maintain algorithm invariants:

- Move the pointer to the right.

```
i++;
```



- Moving from right to left, exchange `a[i]` with each larger entry to its left.

```
for (int j = i; j > 0; j--)
   if (less(a[j], a[j-1]))
         exch(a, j, j-1);
   else break;
```

Mergesort: trace

	a[]															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, aux, 0, 0, 1)	E	M	R	G	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, aux, 2, 2, 3)	E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, aux, 0, 1, 3)	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, aux, 4, 4, 5)	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, aux, 6, 6, 7)	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, aux, 4, 5, 7)	E	G	M	R	E	O	R	S	T	E	X	A	M	P	L	E
merge(a, aux, 0, 3, 7)	E	E	G	M	O	R	R	S	T	E	X	A	M	P	L	E
merge(a, aux, 8, 8, 9)	E	E	G	M	O	R	R	S	E	T	X	A	M	P	L	E
merge(a, aux, 10, 10, 11)	E	E	G	M	O	R	R	S	E	T	A	X	M	P	L	E
merge(a, aux, 8, 9, 11)	E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E
merge(a, aux, 12, 12, 13)	E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E
merge(a, aux, 14, 14, 15)	E	E	G	M	O	R	R	S	A	E	T	X	M	P	E	L
merge(a, aux, 12, 13, 15)	E	E	G	M	O	R	R	S	A	E	T	X	E	L	M	P
merge(a, aux, 8, 11, 15)	E	E	G	M	O	R	R	S	A	E	E	L	M	P	T	X
merge(a, aux, 0, 7, 15)	A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

result after recursive call

Bottom-up mergesort

Basic plan.

- Pass through array, merging subarrays of size 1.
- Repeat for subarrays of size 2, 4, 8, ...

		a[i]														
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E
sz = 1																	
merge(a, aux, 0, 0, 1)		E	M	R	G	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, aux, 2, 2, 3)		E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, aux, 4, 4, 5)		E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, aux, 6, 6, 7)		E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, aux, 8, 8, 9)		E	M	G	R	E	S	O	R	E	T	X	A	M	P	L	E
merge(a, aux, 10, 10, 11)		E	M	G	R	E	S	O	R	E	T	A	X	M	P	L	E
merge(a, aux, 12, 12, 13)		E	M	G	R	E	S	O	R	E	T	A	X	M	P	L	E
merge(a, aux, 14, 14, 15)		E	M	G	R	E	S	O	R	E	T	A	X	M	P	E	L
sz = 2																	
merge(a, aux, 0, 1, 3)		E	G	M	R	E	S	O	R	E	T	A	X	M	P	E	L
merge(a, aux, 4, 5, 7)		E	G	M	R	E	O	R	S	E	T	A	X	M	P	E	L
merge(a, aux, 8, 9, 11)		E	G	M	R	E	O	R	S	A	E	T	X	M	P	E	L
merge(a, aux, 12, 13, 15)		E	G	M	R	E	O	R	S	A	E	T	X	E	L	M	P
sz = 4																	
merge(a, aux, 0, 3, 7)		E	E	G	M	O	R	R	S	A	E	T	X	E	L	M	P
merge(a, aux, 8, 11, 15)		E	E	G	M	O	R	R	S	A	E	E	L	M	P	T	X
sz = 8																	
merge(a, aux, 0, 7, 15)		A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

Yfirlit

Quicksort

- Quicksort er skilvirkt og mikið notað röðunarreiknirit
  - Hefur verið gríðarlega mikið rannsakað síðan þá
  - Kostir og gallar reikniritsins eru vel þekktir
  - Meðal rannsakenda: Robert Sedgewick
- Quicksort fundið upp af Tony Hoare 1960
  - Fann það upp við nám í rússlandi, til að raða orðum
  - Hoare er líka þekktur fyrir framlag til formlegra forritunaraðferða og ALGOL forritunarmálsins

Lýsing

- Lýsing á quicksort:
  1. Veljum safn til að raða og eitthvert stak innan þess til að þjóna sem vendistak (e. *pivot*)
  2. Skiptum upp (e. *partition*) og endurröðum safninu svo að vendistakið sé á réttum stað, engin stök stærri en vendistakið séu í vinstri hluta þess og engin stök minni en vendistakið séu í hægri hluta þess
  3. Notum quicksort endurkvæmt á hvorn hluta fyrir sig
- Skoðum `Quick.java` og glæru 15 í Quicksort

Quicksort trace

	lo	j	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initial values				Q	U	I	C	K	S	O	R	T	E	X	A	M	P	L	E
random shuffle				K	R	A	T	E	L	E	P	U	I	M	Q	C	X	O	S
	0	5	15	E	C	A	I	E	K	L	P	U	T	M	Q	R	X	O	S
	0	3	4	E	C	A	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	2	2	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	0	0	1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	1		1	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	4		4	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	6	6	15	A	C	E	E	I	K	L	P	U	T	M	Q	R	X	O	S
	7	9	15	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	7	7	8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	8		8	A	C	E	E	I	K	L	M	O	P	T	Q	R	X	U	S
	10	13	15	A	C	E	E	I	K	L	M	O	P	S	Q	R	T	U	X
	10	12	12	A	C	E	E	I	K	L	M	O	P	R	Q	S	T	U	X
	10	11	11	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	10		10	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	14	14	15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
	15		15	A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X
result				A	C	E	E	I	K	L	M	O	P	Q	R	S	T	U	X

no partition for subarrays of size 1

Quicksort trace (array contents after each partition)

Skilvirkni

- Besta mögulega tilfelli quicksort - hvert fallskall myndar helmingaskiptingu á safninu
  - Besti samanburðarfjölda má lýsa með $C_N = 2C_{N/2} + N$, svo $C_N \sim N \log N$
- Ástæða fyrir raunverulegri skilvirkni - meðaltilfellið er litlu verra en það besta
  - Ekki nema 39% fleiri samanburðir í "meðaltilfellinu"
  - Má sanna með tölfræðilegum aðferðum
- Í versta tilfellinu - öll stökin lenda á sömu hlið við vendistakið
  - $\sim \frac{N^2}{2}$ samanburðir, en þetta tilvik má venjulega forðast
- Oft betra en sameiningarröðun vegna fárra skiptinga og eiginleika raunverulegra örgjörva
- Skoðum `SortCompare.java`

Endurbætur

- Hægt er að fá meiri hraða úr quicksort með smábreytingum frá `Quick.java`
  - Skipta yfir í innsetningarröðun á smáfylkjum
  - Meðhöndla jafn stór stök sérstaklega
    - Sjá `Quick3way.java`
  - Gott val á vendistaki - t.d. Median-of-3

Hvaða röðunarreiknirit?

Skoðum glæru 48 í Quicksort.

Yfirlit

Forgangsbiðröð

- Forgangsbiðröð (e. *priority queue*) er hugræn gagnagerð
- Almennari gagnagerð en hlaðar og biðraðir, sem við höfum séð áður
  - Í hlaða: Eyðing framkvæmd á því staki sem styst hefur verið á hlaðanum
  - Í biðröð: Eyðing framkvæmd á því staki sem lengst hefur verið í biðröðinni
  - Í forgangsbiðröð: Eyðing framkvæmd á því staki sem hefur hæsta gildið

API

Möguleg skil fyrir forgangsbiðröð:

public class MaxPQ<Key>		
-	`MaxPQ()`	Smiður, býr til tóma forgangsbiðröð
`void`	`insert(Key v)`	Bæta stakinu `key` við f-biðröðina
`Key`	`max()`	Skila stærsta stakinu í biðröðinni
`Key`	`delMax()`	Fjarlægja og skila stærsta stakinu í biðröðinni
`boolean`	`isEmpty()`	Er forgangsbiðröðin tóm?
`int`	`size()`	Fjöldi hluta í forgangsbiðröðinni

Aths: Gætum allt eins skilgreint "minimum priority queue". `Key` þarf að vera samanburðarhæfur.

Skilvirkni forgangsbiðraða

Aðgerðir á forgangsbiðröð af lengd N þurfa tíma sem er háður undirliggjandi gagnagrind

Gagnagrind	Innsetning	Fjarlægja hæsta
Raðað fylki	N	1
Óraðað fylki	1	N
Hrúga	$\log N$	$\log N$
Einhyrningur[1]	1	1

Það að nota óraðað fylki er "löt" aðferðafræði, það að nota raðað fylki "áköf" aðferðafræði.

[1] Einhyrningar eru ekki til

Yfirlit

Hrúgur

Hrúga (e. *heap*) er tré sem uppfyllir hrúguskilyrði (e. *heap property*). Við notum fylki til að geyma hrúgur.
Skoðum glærur 16-17, 20-23 í PriorityQueues.

Binary heap representations

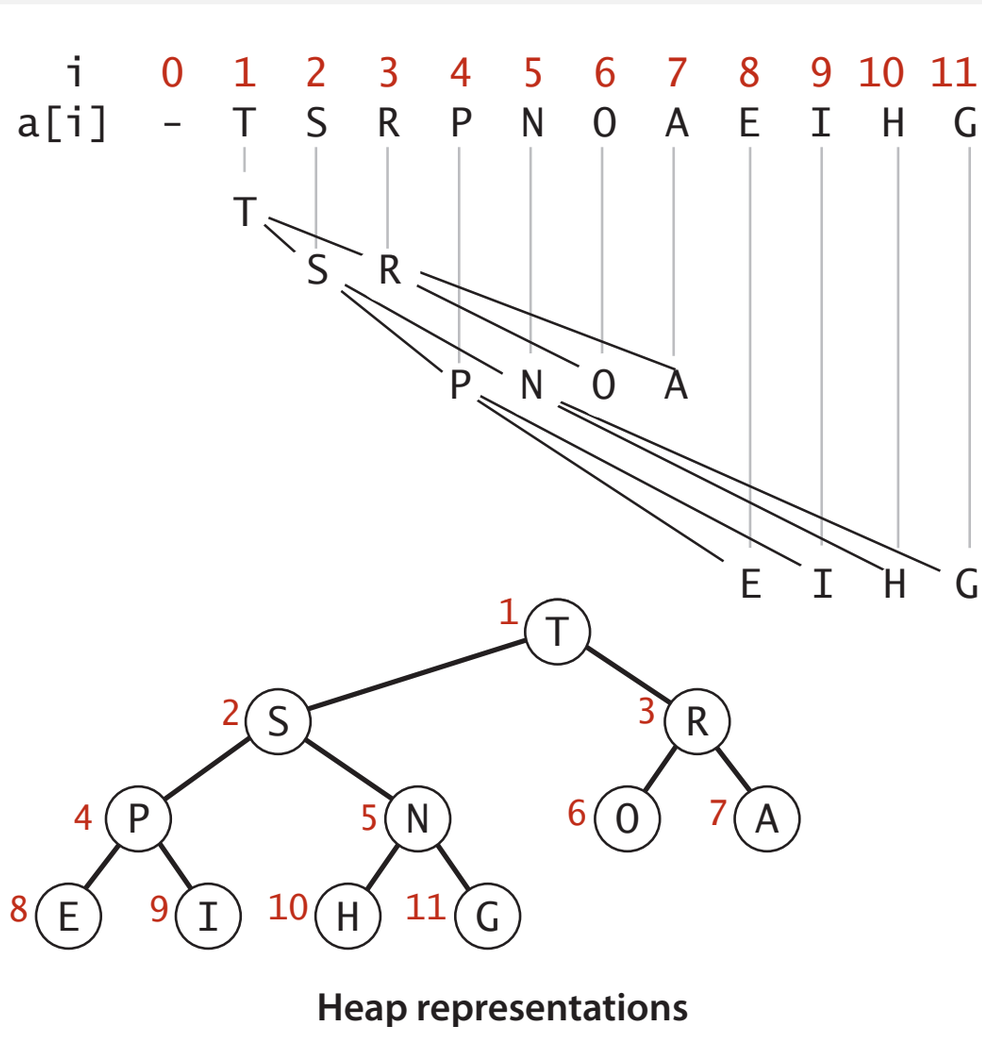
Binary heap. Array representation of a heap-ordered complete binary tree.

Heap-ordered binary tree.

- Keys in nodes.
- Parent's key no smaller than children's keys.

Array representation.

- Indices start at 1.
- Take nodes in level order.
- No explicit links needed!



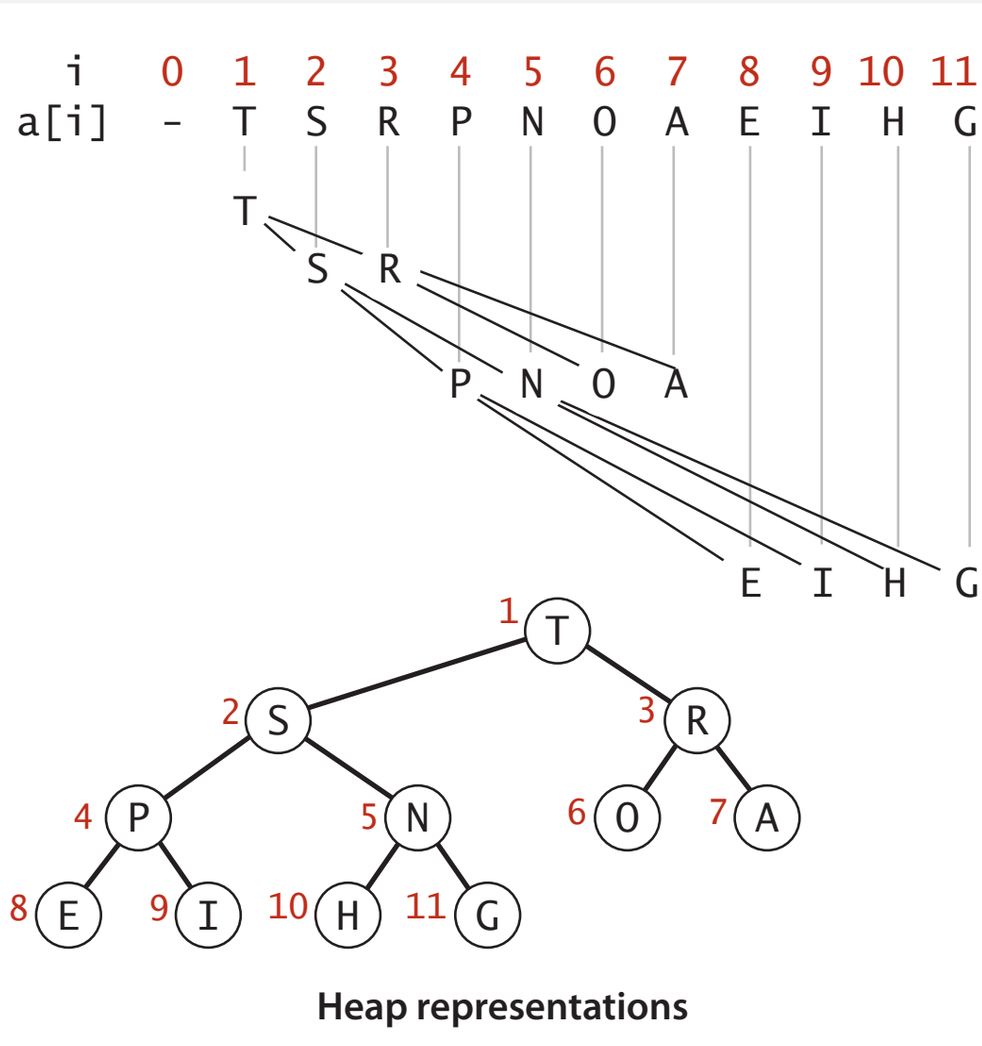
Heap representations

Binary heap properties

Proposition. Largest key is `a[1]`, which is root of binary tree.

Proposition. Can use array indices to move through tree.

- Parent of node at `k` is at `k/2`.
- Children of node at `k` are at `2k` and `2k+1`.



Heap representations

Promotion in a heap

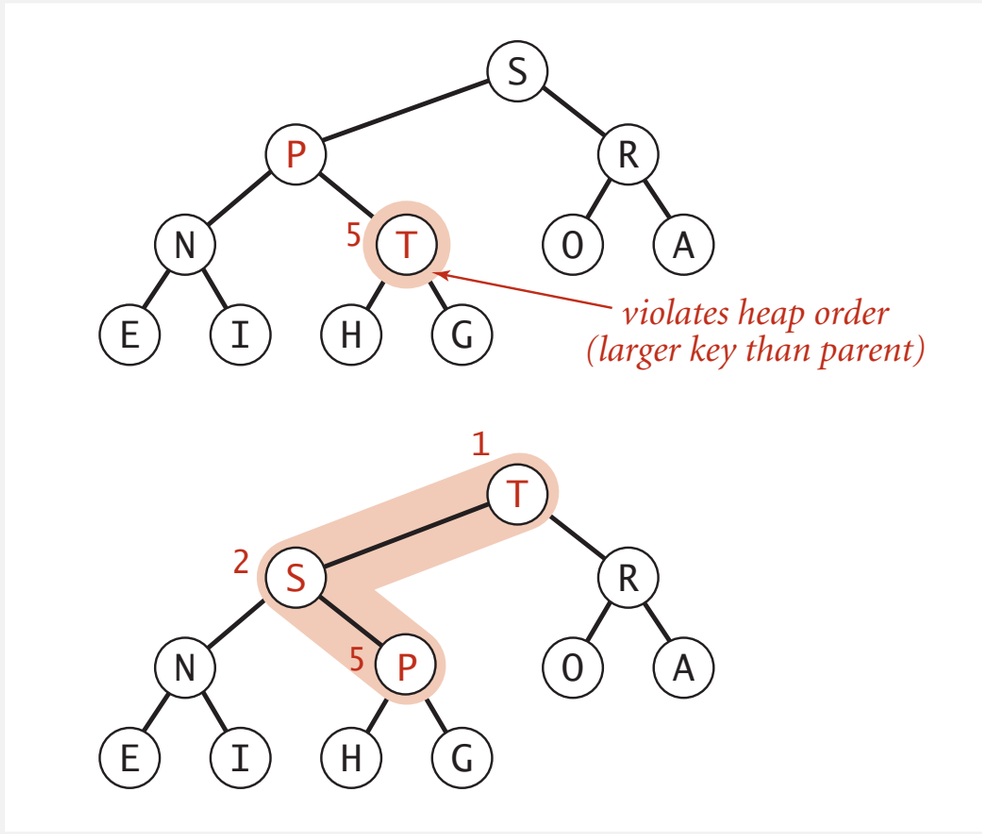
Scenario. Child's key becomes larger key than its parent's key.

To eliminate the violation:

- Exchange key in child with key in parent.
- Repeat until heap order restored.

```
private void swim(int k)
{
   while (k > 1 && less(k/2, k))
   {
      exch(k, k/2);
      k = k/2;
   }
}
```

parent of node at k is at k/2



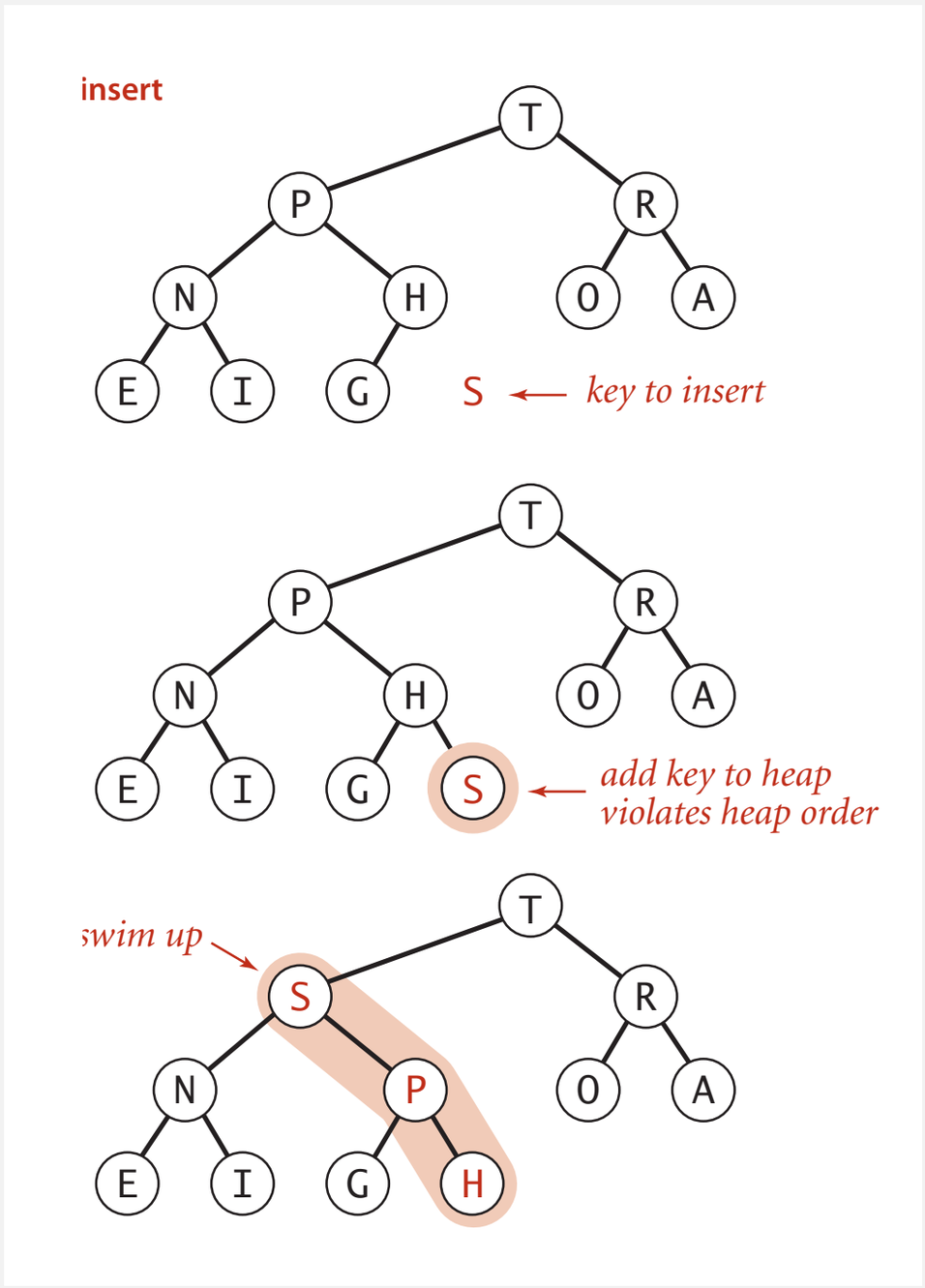
Peter principle. Node promoted to level of incompetence.

Insertion in a heap

Insert. Add node at end, then swim it up.

Cost. At most $1 + \lg N$ compares.

```
public void insert(Key x)
{
   pq[++N] = x;
   swim(N);
}
```



Demotion in a heap

Scenario. Parent's key becomes smaller than one (or both) of its children's.

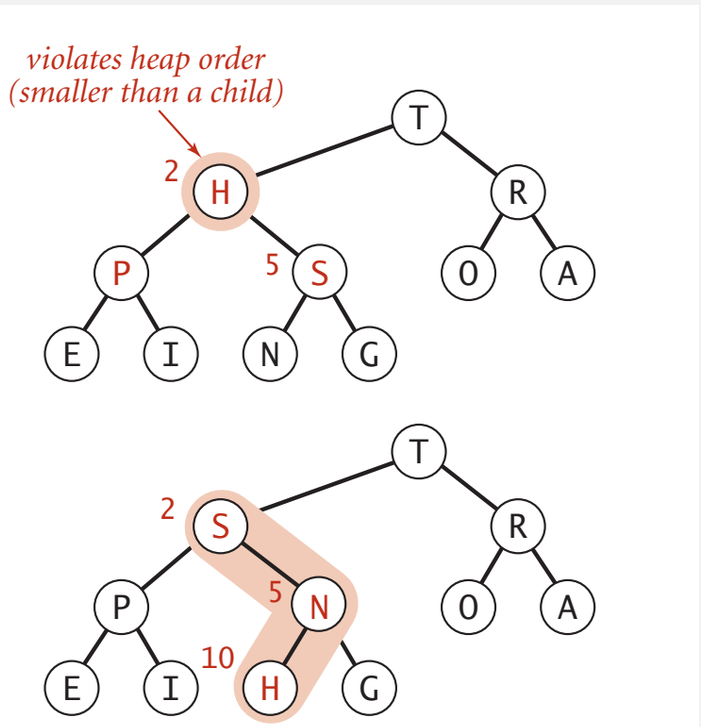
To eliminate the violation:

why not smaller child?

- Exchange key in parent with key in larger child.
- Repeat until heap order restored.

```
private void sink(int k)
{
   while (2*k <= N)
   {
      int j = 2*k;
      if (j < N && less(j, j+1)) j++;
      if (!less(k, j)) break;
      exch(k, j);
      k = j;
   }
}
```

children of node at k are 2k and 2k+1



Top-down reheapify (sink)

Power struggle. Better subordinate promoted.

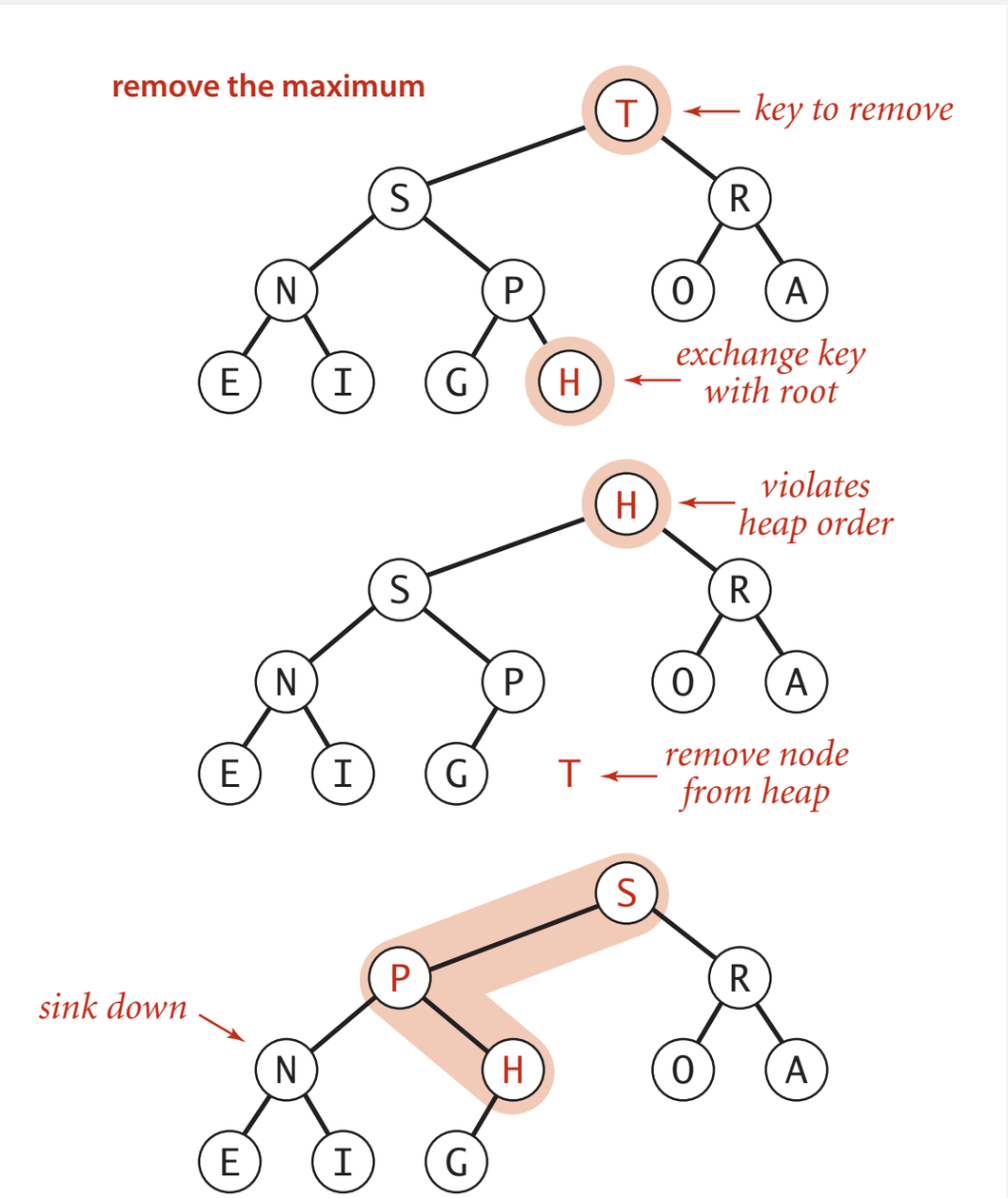
Delete the maximum in a heap

Delete max. Exchange root with node at end, then sink it down.

Cost. At most $2 \lg N$ compares.

```
public Key delMax()
{
   Key max = pq[1];
   exch(1, N--);
   sink(1);
   pq[N+1] = null;
   return max;
}
```

prevent loitering



Tími

- Athugum - um fullskipuð tré er að ræða
- Innsetning og eyðing felur í sér færslu á milli hæða í trénu
- Fjöldi aðgerða takmarkast af hæð trésins
  - Hæð fullskipaðs trés með N hnútum er $\lfloor \log_2 n \rfloor$
- Praktísk vandræði - langt á milli staka, hentar skyndiminni örgjörva illa

Þessi glærupakki

Tengill á fyrirlestraræfingu:
https://goo.gl/forms/b4uyD6Z17qfsevas1

`SortCompare.java` má finna á Github.
Kóða fyrir algs4 reiknirit má finna á
http://algs4.cs.princeton.edu/code/