Comments about take-home exam

1. In general, much better.

Comments about take-home exam

1. In general, much better.
2. If it was not approved, do the re-exam.
   No retries on the re-exam.

Comments about take-home exam

1. In general, much better.
2. If it was not approved, do the re-exam.
   No retries on the re-exam.
3. It is due at 8:15 on November 2.

Comments about take-home exam

1. In general, much better.
2. If it was not approved, do the re-exam.
   No retries on the re-exam.
3. It is due at 8:15 on November 2.
4. Read all comments, even if it was approved.

New assignment

- Assignment 3 is available.

New assignment

- Assignment 3 is available.
- It is due at 8:15 on November 5.

Comments about first assignment

If it was not approved:

1. You need to do the redo, by 8:30 on October 29.

Comments about first assignment

If it was not approved:

1. You need to do the redo, by 8:30 on October 29.
2. Fix all problems with your original assignment.

Comments about first assignment

If it was not approved:

1. You need to do the redo, by 8:30 on October 29.
2. Fix all problems with your original assignment.
3. Turn in new version via Blackboard and graded version to your "instruktor".
4. You will only be allowed a redo on one more assignment.

Encouragement to study

http://www.tv2fyn.dk/arkiv/2015/10/14?video_id=85676&autoplay=1

Informal course evaluation

Is this course giving you a good overview of what computer science is?

A. Very useful.
B. Somewhat useful.
C. I needed an overview, but this course is not giving it.
D. I did not need an overview, but it is still good.
E. I did not need an overview and do not want one.

Vote at m.socrative.com. Room number 415439.

Informal course evaluation

What is your opinion about the pace of the lectures?

A. Much too fast.
B. A little too fast.
C. Close to right.
D. A little too slow.
E. Much too slow.

Vote at m.socrative.com. Room number 415439.

Informal course evaluation

How difficult is the course?

A. Much too difficult.
B. A little too difficult.
C. A good level.
D. A little too easy.
E. Much too easy.

Vote at m.socrative.com. Room number 415439.

Informal course evaluation

What do you think about U50A?

A. It's a terrible lecture hall.
B. It's still bad, but better with screens and microphone.
C. It's OK.
D. It's good now.

Vote at m.socrative.com. Room number 415439.

Informal course evaluation

1. What do you like about the course?

Informal course evaluation

1. What do you like about the course?
2. What can be improved?

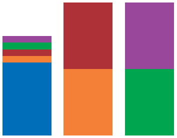
Informal course evaluation

1. What do you like about the course?
2. What can be improved?
3. Any comments I should give your “instruktor”s?

Classical bin packing

Use as few bins as possible:
Item sizes: $n \times [1/2, \epsilon]$
Bin size: 1

Result by First-Fit algorithm:



Dual bin packing

Given a fixed number of bins, pack as many items as possible.

Bin size: 1
Number of bins: 4
Item sizes:

- $\frac{1}{4}$, $\frac{1}{4}$, $\frac{1}{4}$
- $\frac{5}{12}$, $\frac{1}{3}$
- $\frac{5}{12}$, $\frac{1}{3}$
- $\frac{5}{12}$, $\frac{1}{3}$
- $\frac{1}{3}$, $\frac{1}{3}$, $\frac{1}{3}$

Can they all be there?

Bin packing

First-Fit is an on-line algorithm:

It handles requests without looking at future requests.

Solving bin packing optimally is NP-hard.

Brute force takes a long time.

Approximation algorithms: First-Fit-Decreasing, even better...

Special case: all sizes multiples of $\frac{1}{12}$.

Fill one bin completely if possible.

First-Fit for dual bin packing

```
procedure First-Fit-Dual(List):
{ Input: List is a list of items with sizes ≤ 1 }
{ Output: Number of rejected items }

    k := number of bins { all empty }
    Count := 0 { number rejected }
        get next item x and remove from list
        i := 1
        while (i ≤ k and x does not fit in bin i)
            i := i + 1
        if (i ≤ k)
            then put x in bin i
            else Count := Count+1
    return(Count)
```

First-Fit for dual bin packing (correct)

```
procedure First-Fit-Dual(List):
{ Input: List is a list of items with sizes ≤ 1 }
{ Output: Number of rejected items }

    k := number of bins { all empty }
    Count := 0 { number rejected }
    while there are still items in the list
    begin
        get next item x and remove from list
        i := 1
        while (i ≤ k and x does not fit in bin i)
            i := i + 1
        if (i ≤ k)
            then put x in bin i
            else Count := Count+1
    end
    return(Count)
```

File access

2 standard methods for accessing data:

- sequential access
- random access: access via index or ID (key) for data element

Questions

1. What can be done using only Sequential access?
2. How can one implement Random access?

Merge Sort

procedure MergeSort(A, f, l):
{ Input: Array A with first index f and last index l }
{ Output: Sorted array, A, with same entries as input A }

if ($f < l$) **then**
$m := (f + l)$ div 2
MergeSort(A, f, m)
MergeSort(A, $m + 1$, l)
MergeArrays($A[f..m]$, $A[m + 1..l]$, C)
Copy C to A

MergeSort(A, 1, length(A));

Analysis of Merge Sort

Let $T(n)$ be the maximum number of comparisons MergeSort uses if length$(A) = n$.

$$\begin{aligned}
T(n) &\leq T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + M\left(\left\lceil \frac{n}{2} \right\rceil, \left\lfloor \frac{n}{2} \right\rfloor\right) \\
&\leq T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor - 1\right) \\
&\leq T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n - 1
\end{aligned}$$

$T(n) \in \Theta(n \log n)$.

Analysis of Merge Sort

$$T(n) \leq T\left(\left\lceil\frac{n}{2}\right\rceil\right) + T\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + n - 1$$

Prove by induction: $T(n) \leq n\log_2(n)$, if $n = 2^j$ for some integer j

Base case: $n = 1$. $1 \cdot \log_2(1) = 0 = T(1)$.

Induction hypothesis: For all $k < n$, where $k = 2^i$,
$T(k) \leq k\log_2(k)$.

Induction step (prove for n):

$$\begin{aligned}
T(n) &\leq T\left(\left\lceil\frac{n}{2}\right\rceil\right) + T\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + n - 1 \\
&\leq 2T\left(\frac{n}{2}\right) + n - 1 \\
&\leq 2\frac{n}{2}\log_2\left(\frac{n}{2}\right) + n - 1 \\
&\leq n(\log_2 n - 1) + n - 1 \\
&\leq n\log_2 n
\end{aligned}$$

Analysis of Merge Sort

$T(n) \leq n \log_2(n)$, if $n = 2^j$ for some integer j.

If $n \neq 2^j$ for any integer j, $T(n) \leq T(n')$ where n' is the next power of 2 larger than n.

In general $T(n) \leq (2n) \log_2(2n) \leq 2n \log_2 n + 2n$.

So $T(n) \in \Theta(n \log n)$.

Merging more than 2 lists

Problem:

Input: 3 lists, A, B and C are sorted.

Output: 1 sorted list, D, containing the entries of $A \cap B \cap C$.

Intersecting 3 lists

Input: 3 lists, A, B and C are sorted.

Output: 1 sorted list, D, containing the entries of $A \cap B \cap C$.

Merge Step:

- Compare current records of A, B and C.
- If all the same, put record in D. Advance to next record in all of A, B and C.
- If current in A is smaller than current in either B or C, advance to next record in A. (Do same for B and C.)

```
procedure MergeFiles(A, B, C):
    open(A); open(B); open(C); fA,fB,fC := false;
    if (isEndOfFile(A) and isEndOfFile(B)) then Stop with C empty
    if (not isEndOfFile(A)) then currentA := readNext(A); fA:= true;
    if (not isEndOfFile(B)) then currentB := readNext(B); fB := true;
    while (fA and fB) do
        if (currentA ≤ currentB) then
            writeNext(currentA, C)
            if (not isEndOfFile(A)) then currentA := readNext(A)
            else fA := false
        else
            writeNext(currentB, C)
            if (not isEndOfFile(B)) then currentB := readNext(B)
            else fB := false
    Starting with the current record in the input file which is not at EOF
        copy the remaining records to C
    close(A); close(B); close(C)
```

Random access API

random access: access via ID (key) for data element

Operations:

findElm(ID)

insertElm(ID,elementData)

deleteElm(ID)

open()

close()

Examples:

- dictionaries in Python
- arrays in Java — with ID = index in array