Comments about take-home exam

1. Some assignments were not approved, even though the TA commented that it was very good.

Comments about take-home exam

1. Some assignments were not approved, even though the TA commented that it was very good.
2. Come ask if there's something you don't understand, such as "in general".

Comments about take-home exam

1. Some assignments were not approved, even though the TA commented that it was very good.
2. Come ask if there's something you don't understand, such as "in general".
3. Look at your notes from discussion section. Sometimes problems are similar.

Comments about take-home exam

1. Some assignments were not approved, even though the TA commented that it was very good.
2. Come ask if there's something you don't understand, such as "in general".
3. Look at your notes from discussion section. Sometimes problems are similar.
4. Look in your textbook or my slides (or notes on the course's homepage) for how to do things (page 239 for `repeat` loop, for example).

Comments about take-home exam

1. Some assignments were not approved, even though the TA commented that it was very good.
2. Come ask if there's something you don't understand, such as "in general".
3. Look at your notes from discussion section. Sometimes problems are similar.
4. Look in your textbook or my slides (or notes on the course's homepage) for how to do things (page 239 for `repeat` loop, for example).
5. Don't write more than necessary, but always explain, so it's easy to see that your answer is correct and you understand it.

Comments about take-home exam

1. Some assignments were not approved, even though the TA commented that it was very good.
2. Come ask if there's something you don't understand, such as "in general".
3. Look at your notes from discussion section. Sometimes problems are similar.
4. Look in your textbook or my slides (or notes on the course's homepage) for how to do things (page 239 for `repeat` loop, for example).
5. Don't write more than necessary, but always explain, so it's easy to see that your answer is correct and you understand it.
6. If given a choice, don't choose to do the hard problem wrong.

Comments about take-home exam

1. Some assignments were not approved, even though the TA commented that it was very good.
2. Come ask if there's something you don't understand, such as "in general".
3. Look at your notes from discussion section. Sometimes problems are similar.
4. Look in your textbook or my slides (or notes on the course's homepage) for how to do things (page 239 for `repeat` loop, for example).
5. Don't write more than necessary, but always explain, so it's easy to see that your answer is correct and you understand it.
6. If given a choice, don't choose to do the hard problem wrong.
7. Read all comments, even if it was approved. Correct all errors if it was not.

Comments about take-home exam

1. Some assignments were not approved, even though the TA commented that it was very good.
2. Come ask if there's something you don't understand, such as "in general".
3. Look at your notes from discussion section. Sometimes problems are similar.
4. Look in your textbook or my slides (or notes on the course's homepage) for how to do things (page 239 for `repeat` loop, for example).
5. Don't write more than necessary, but always explain, so it's easy to see that your answer is correct and you understand it.
6. If given a choice, don't choose to do the hard problem wrong.
7. Read all comments, even if it was approved. Correct all errors if it was not.
8. Submit redo via Blackboard. Give your TA your correct original.

Sorting

How do you sort? Think about cards.

Insertion Sort

```
procedure Sort(List):
{ Input: List is a list }
{ Output: List, with same entries, but in nondecreasing order }

    N := 2
    while (N ≤ length(List))
    begin
        Pivot := Nth entry
        j := N − 1
        while (j > 0 and jth entry > Pivot)
        begin
            move jth entry to loc. j + 1
            j := j − 1
        end
        place Pivot in j + 1st loc.
        N := N + 1
    end
```

Insertion Sort

What happens if List has 0 or 1 entry?

A. Sort crashes
B. Sort returns the input list unchanged
C. Sort returns something wrong

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

Insertion Sort

17	8	15	53	18	12	2	75
1	2	3	4	5	6	7	8

N: 2

Pivot: 8

j: 1

jth entry: 17

Insertion Sort

8	17	15	53	18	12	2	75
1	2	3	4	5	6	7	8

N: 2

Pivot: 8

j: 0

jth entry: none

Insertion Sort

8	17	15	53	3	12	2	75
1	2	3	4	5	6	7	8

N: 3

Pivot: 15

j: 2

jth entry: 17

Insertion Sort

8	17	15	53	3	12	2	75
1	2	3	4	5	6	7	8

N: 3

Pivot: 15

j: 2

jth entry: 17

Continue on board.

Insertion Sort — correctness

```
procedure Sort(List):
{ Input: List is a list }
{ Output: List, with same entries, but in nondecreasing order }
    N := 2
    while (N ≤ length(List)
    begin
{ loop invariants:
    1. entries 1 thru N − 1 in List are in sorted order
    2. the same items are in List as originally }
        Pivot := Nth entry
        j := N − 1
        while (j > 0 and jth entry > Pivot) begin
            move jth entry to loc. j + 1
            j := j − 1        end
        place Pivot in j + 1st loc.
        N := N + 1
    end
```

Insertion Sort — correctness

```
procedure Sort(List):
{ Input: List is a list }
{ Output: List, with same entries, but in nondecreasing order }
    N := 2
    while (N ≤ length(List) begin
        Pivot := Nth entry
        j := N − 1
        while (j > 0 and jth entry > Pivot) begin
{ loop invariants: 1. no item in loc. j + 1
    2. entries in locs. j + 2 to N are larger than Pivot
    3. entries in locs. 1 to N − 1 stay in same relative order
    4. no entries in locs. N + 1 to length(List) are changed }
            move jth entry to loc. j + 1
            j := j − 1   end
        place Pivot in j + 1st loc.
        N := N + 1
    end
```

Insertion Sort — analysis

Suppose list has n entries.

How many comparisons occur in the best case?

A. 1
B. 2
C. n-1
D. n
E. n+1

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

(What is the best case?)

Insertion Sort — analysis

Worst case number of comparisons:

Outer loop from 2 to n.

Insertion Sort

```
procedure Sort(List):
{ Input: List is a list }
{ Output: List, with same entries, but in nondecreasing order }

    N := 2
    while (N ≤ length(List)
    begin
        Pivot := Nth entry
        j := N − 1
        while (j > 0 and jth entry > Pivot)
        begin
            move jth entry to loc. j + 1
            j := j − 1
        end
        place Pivot in j + 1st loc.
        N := N + 1
    end
```

Insertion Sort — analysis

Worst case number of comparisons:

Outer loop from 2 to n.

Inner loop from $N - 1$ to 1.

Total: $\sum_{N=2}^{n}(N-1) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \in \Theta(n^2)$.

Insertion Sort — analysis

Worst case number of comparisons:

Outer loop from 2 to n.

Inner loop from $N - 1$ to 1.

Total: $\sum_{N=2}^{n}(N-1) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \in \Theta(n^2)$.

Can it take this many comparisons?

Insertion Sort — analysis

Worst case number of comparisons:

Outer loop from 2 to n.

Inner loop from $N-1$ to 1.

Total: $\sum_{N=2}^{n}(N-1) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \in \Theta(n^2)$.

Can it take this many comparisons?

Yes.

Insertion Sort — analysis

What list gives this worst case?

A. an ordered list
B. a list in reverse order
C. a random list
D. none of the above
E. all of the above

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

Insertion Sort — analysis

Worst case number of comparisons:

Outer loop from 2 to n.

Inner loop from $N-1$ to 1.

Total: $\sum_{N=2}^{n}(N-1) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \in \Theta(n^2)$.

Average case number of comparisons:

On average place next Pivot half way down the list.

$\frac{n(n-1)}{4} \in \Theta(n^2)$.

Insertion Sort — analysis

Worst case number of comparisons:

Outer loop from 2 to n.

Inner loop from $N - 1$ to 1.

Total: $\sum_{N=2}^{n}(N-1) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \in \Theta(n^2)$.

Average case number of comparisons:

On average place next Pivot half way down the list.

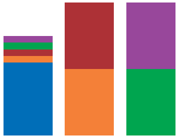
$\frac{n(n-1)}{4} \in \Theta(n^2)$.

There exist algorithms which do $\Theta(n \log n)$ comparisons.

Classical bin packing

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

Result by First-Fit algorithm:



Classical bin packing

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

Result by Worst-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?

Dual bin packing

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?

First check:

$$3\frac{3}{12} + 3\frac{9}{12} + 3\frac{4}{12} = 4$$

Dual bin packing

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?

What about First-Fit?

An optimal algorithm?

Bin packing

First-Fit is an on-line algorithm:

It handles requests without looking at future requests.

Some problems are on-line in nature. Examples?

Solving bin packing optimally is NP-hard.

Brute force takes a long time.

Bin packing

First-Fit is an on-line algorithm:

It handles requests without looking at future requests.

Some problems are on-line in nature. Examples?

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)
```