Lecture 24 - Dec. 3

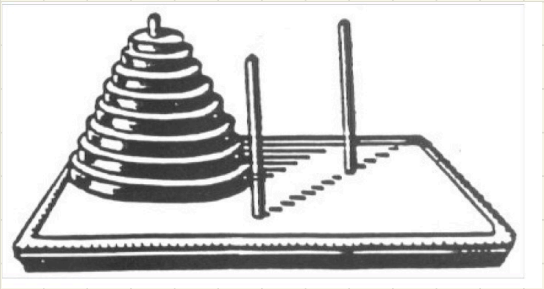
Recursion

Tower of Hanoi: Specification, Legend
Tower of Hanoi: Java, Tracing
Tower of Hanoi: Running Time

Announcements/Reminders

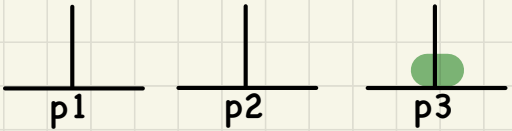
- **Lab5** due **midnight** today
  + Required study: **Abstract Classes** & **Interfaces**
- **ProgTest3** results released
- Extra office hours: 3pm to 5pm on Thursday
- **Exam** Review Session (Zoom): 3pm on Friday
- Materials for tutorial session on **recursion**

Tower of Hanoi: Strategy



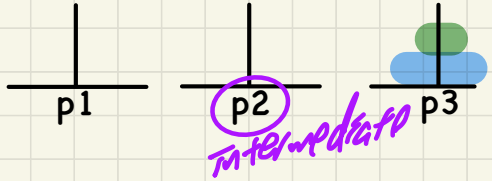
Consider 1 disk: A

move  from p1 to p3



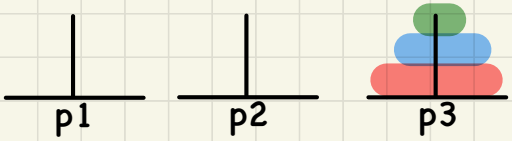
Consider 2 disks: A < B

move  from p1 to p3

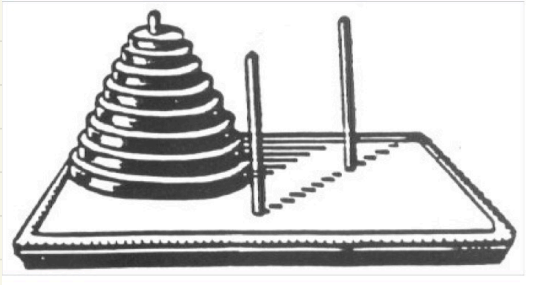


Consider 3 disks: A < B < C

move  from p1 to p3



Tower of Hanoi: Strategy

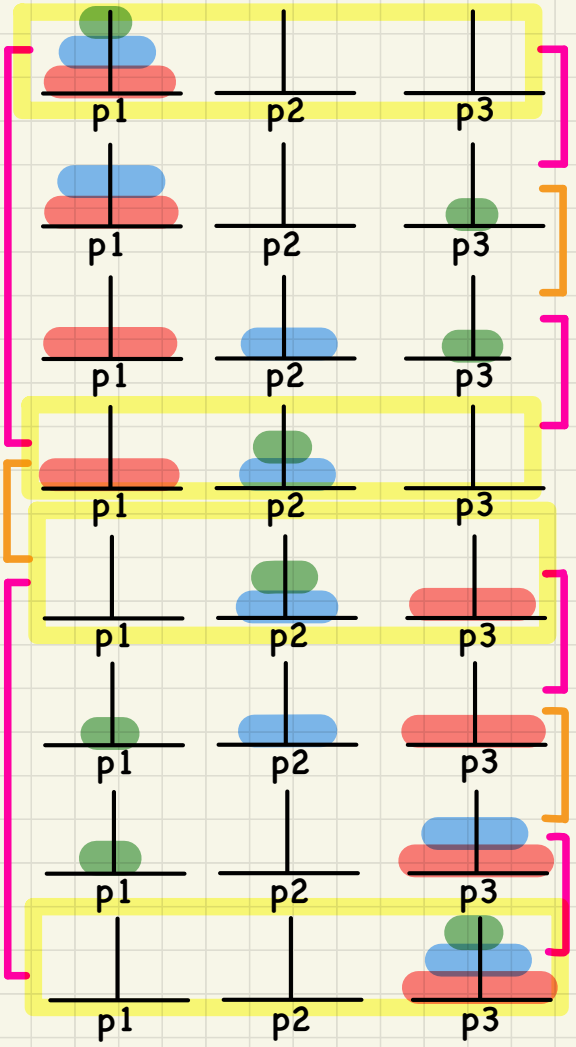


Consider 3 disks A < B < C

move  from p1 to p3

move  from p1 to p2

move  from p2 to p3



move  from p1 to p3

move  from p1 to p2

move  from p3 to p2

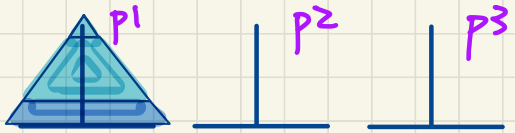
move  from p1 to p3

move  from p2 to p1

move  from p2 to p3

move  from p1 to p3

Tower of Honoi in Java



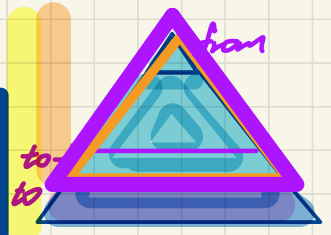

```
void towerOfHanoi(String[] disks) {
  tohHelper(disks, 0, disks.length - 1, 1, 3);
}
void tohHelper(String[] disks, int from, int to, int ori, int des){
  if(from > to) { }
  else if(from == to) {
    print("move " + disks[to] + " from " + ori + " to " + des);
  }
  else {
    int intermediate = 6 - ori - des;
    tohHelper(disks, from, to - 1, ori, intermediate);
    print("move " + disks[to] + " from " + ori + " to " + des);
    tohHelper(disks, from, to - 1, intermediate, des);
  }
}
```

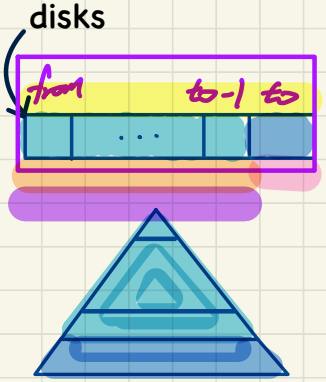
recursive helper method

pegs

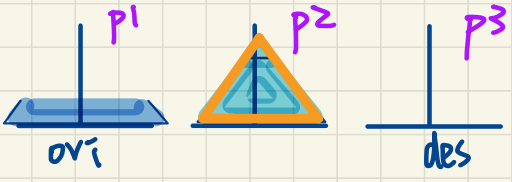
only one disk to move

1 2 2 1. 3 3

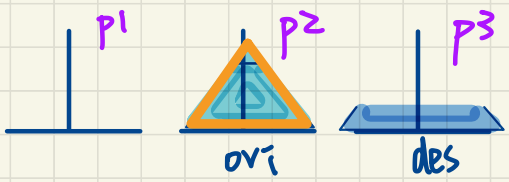




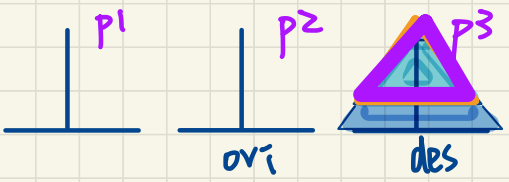
Stage 1



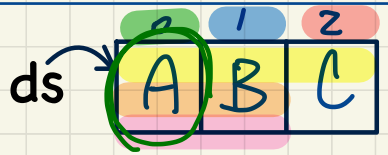
Stage 2

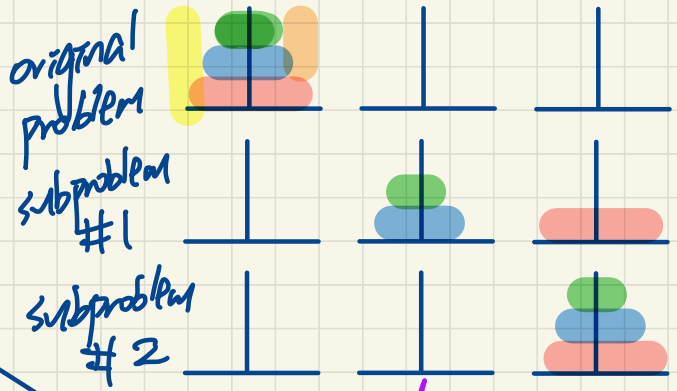


Stage 3



Tower of Hanoi: Tracing





toH(ds, 0, 2, p1, p3) (ori: p1, des.: p3)

toH(ds, 0, 1, p1, p2) (intermediate: p2)

④ move C : p1 to p3

toH(ds, 0, 1, p2, p3) (intermediate: p2, des: p3)

① toH(ds, 0, 0, p1, p3) → move A : p1 to p3

② move B : p1 to p2

③ toH(ds, 0, 0, p3, p2) → move A : p3 to p2

⑤

⑥

⑦

Exercise

Tower of Hanoi: Tracing

Say *ds* (disks) is $\{A, B, C\}$, where $A < B < C$.

$$
tohH(ds, \underbrace{0,2}_{\{A,B,C\}}, p1, p3) = \begin{cases}
tohH(ds, \underbrace{0,1}_{\{A,B\}}, p1, p2) = \begin{cases}
tohH(ds, \underbrace{0,0}_{\{A\}}, p1, p3) = \{ \texttt{Move A: p1 to p3} \\
\texttt{Move B: p1 to p2} \\
tohH(ds, \underbrace{0,0}_{\{A\}}, p3, p2) = \{ \texttt{Move A: p3 to p2}
\end{cases} \\
\texttt{Move C: p1 to p3} \\
tohH(ds, \underbrace{0,1}_{\{A,B\}}, p2, p3) = \begin{cases}
tohH(ds, \underbrace{0,0}_{\{A\}}, p2, p1) = \{ \texttt{Move A: p2 to p1} \\
\texttt{Move B: p2 to p3} \\
tohH(ds, \underbrace{0,0}_{\{A\}}, p1, p3) = \{ \texttt{Move A: p1 to p3}
\end{cases}
\end{cases}
$$

Tower of Hanoi: Running Time

```
void towerOfHanoi(String[] disks) {
  tohHelper(disks, 0, disks.length - 1, 1, 3);
}
void tohHelper(String[] disks, int from, int to, int ori, int des){
  if(from > to) { }
  else if(from == to) {
    print("move " + disks[to] + " from " + ori + " to " + des);
  }
  else {
    int intermediate = 6 - ori - des;
    tohHelper(disks, from, to - 1, ori, intermediate);
    print("move " + disks[to] + " from " + ori + " to " + des);
    tohHelper(disks, from, to - 1, intermediate, des);
  }
}
```

T(n)

T(n-1)

1

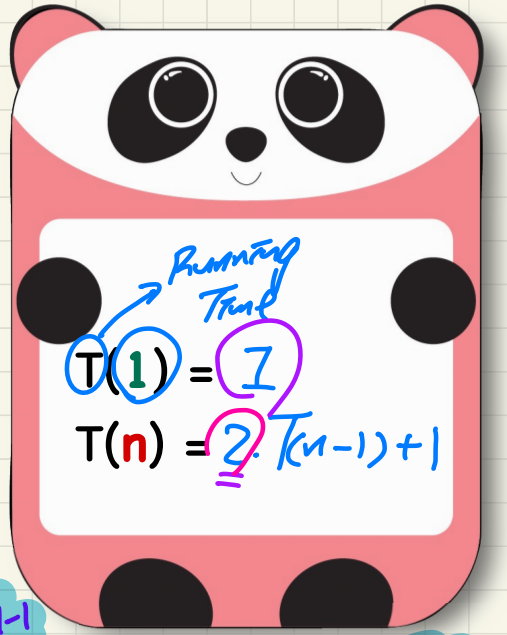
T(n-1)

n - ? = 1

(n-1)

unfolding

Running Time as a Recurrence Relation



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

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

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

$+3 = 2 \cdot (2 \cdot 2 \cdots T(1) \cdots + 1) + 1) + \cdots 1)$

$2^{n-1} + (n-1)$

2^n

2^3