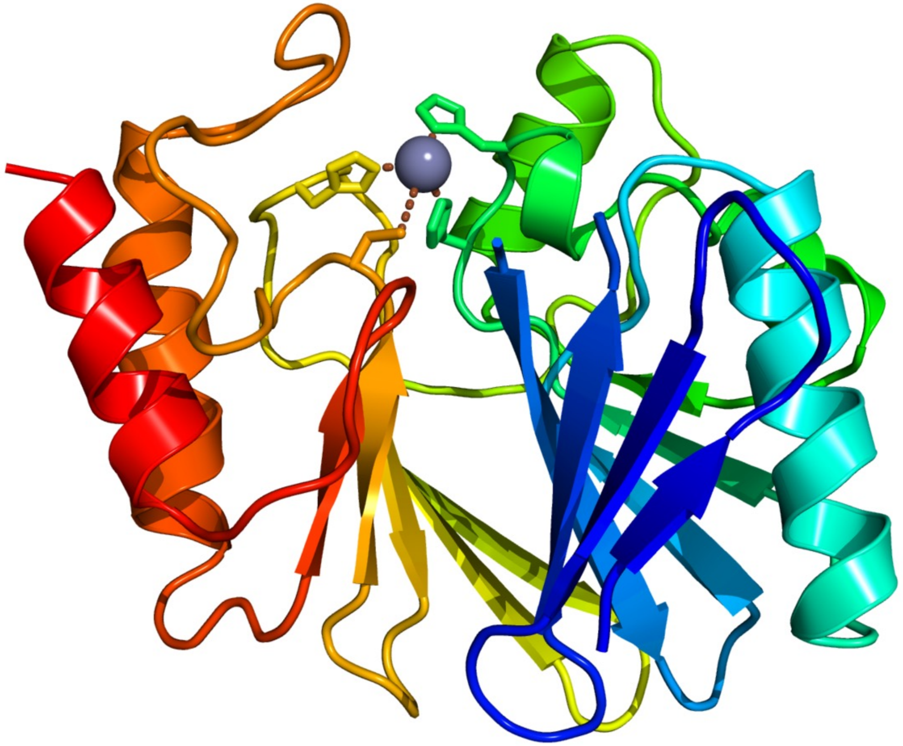
Lecture 13: Recursive Backtracking I

07/25/22

Reminders

- A3 Resubmission due Wednesday 7/27 @ 11:59pm
- A5 due Thursday 7/28 @ 11:59pm

A new type of problem to solve



fourAB

Write a method `fourAB` that prints out all strings of length 4 composed only of a’s and b’s.

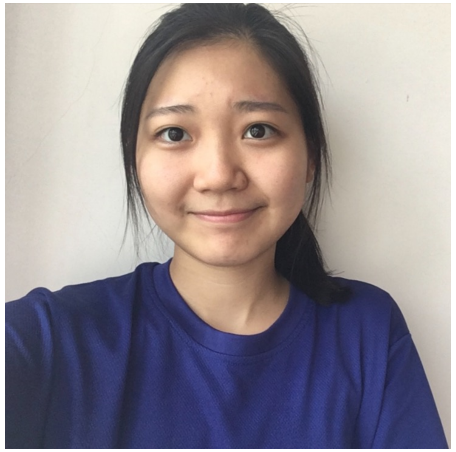
```
aaaa     baaa
aaab     baab
aaba     baba
aabb     babb
abaa     bbaa
abab     bbab
abba     bbba
abbb     bbbb
```

fourAB

Yafqa

a / b

Anju

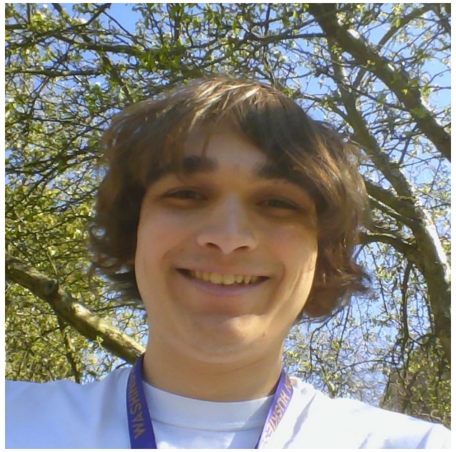


a / b

Mia

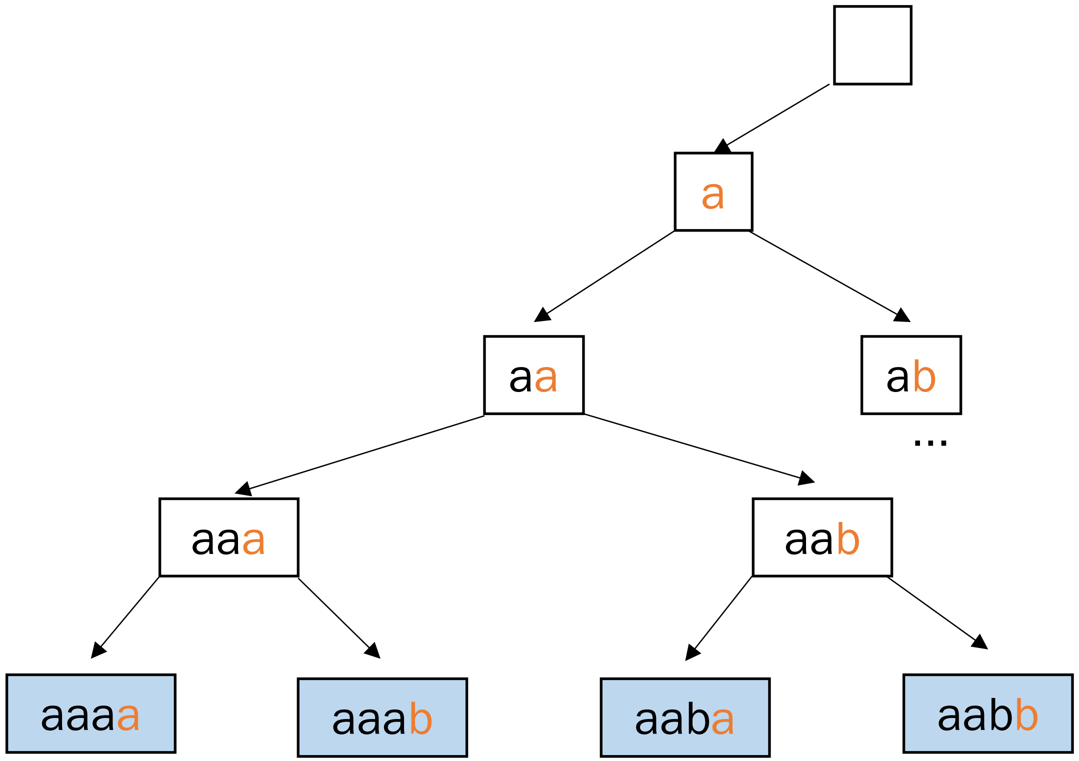
a / b

Raymond



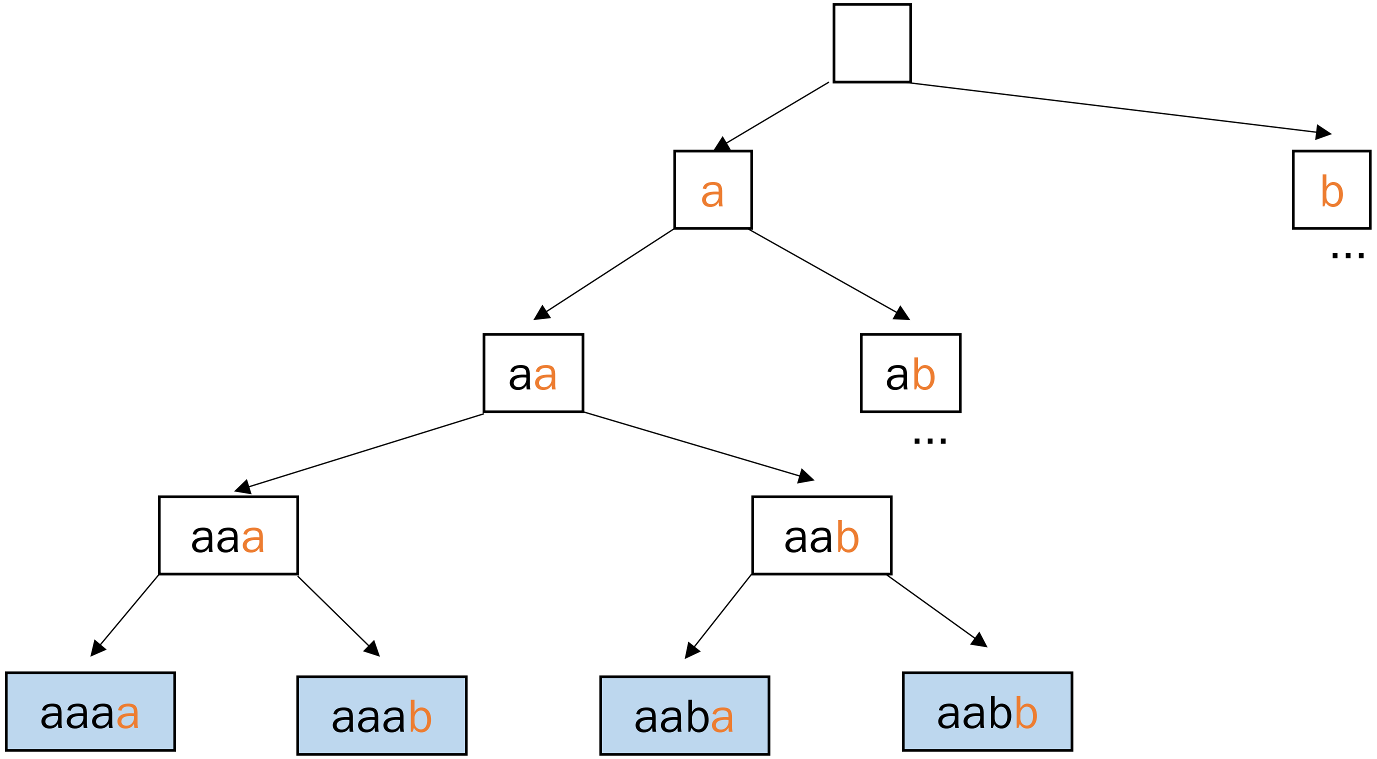
a / b

Decision Tree – fourAB



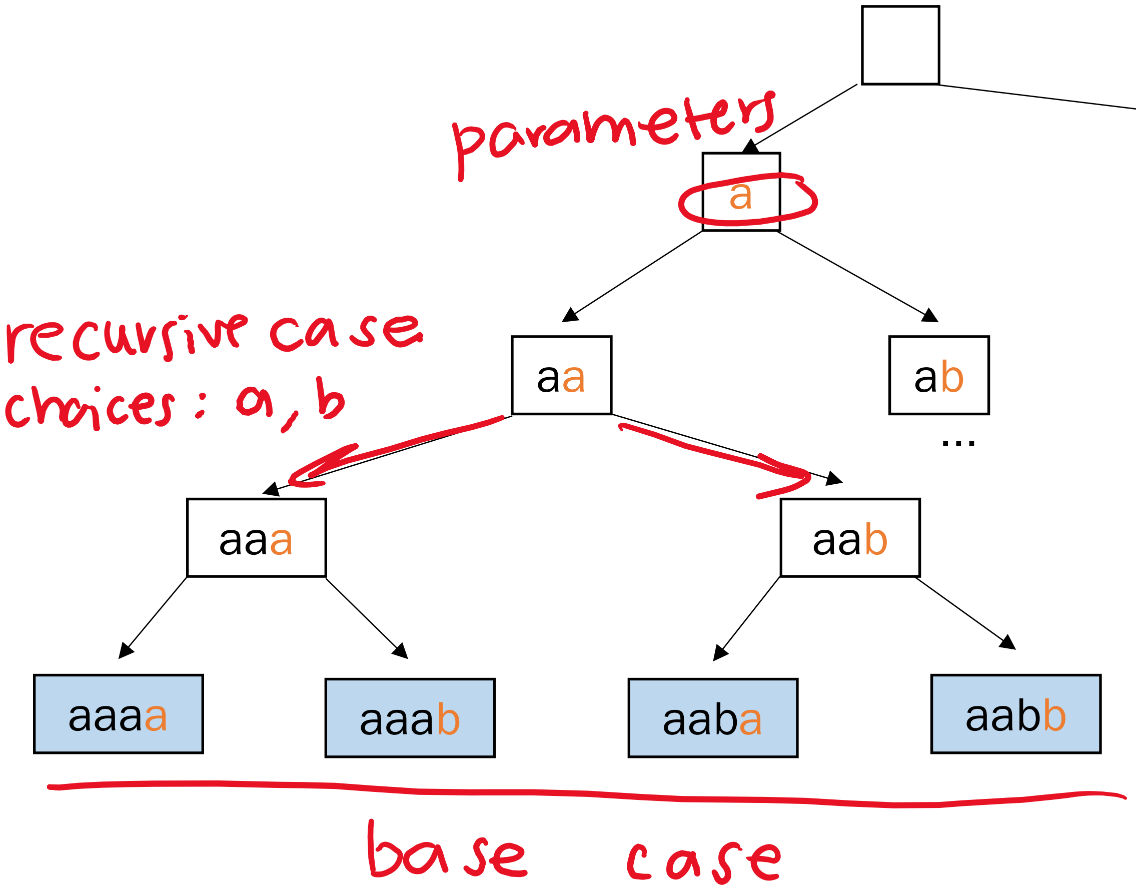
Output:
aaaa
aaab
aaba
aabb

Decision Tree – fourAB



Output:
aaaa
aaab
aaba
aabb

Decision Tree – fourAB



Output:

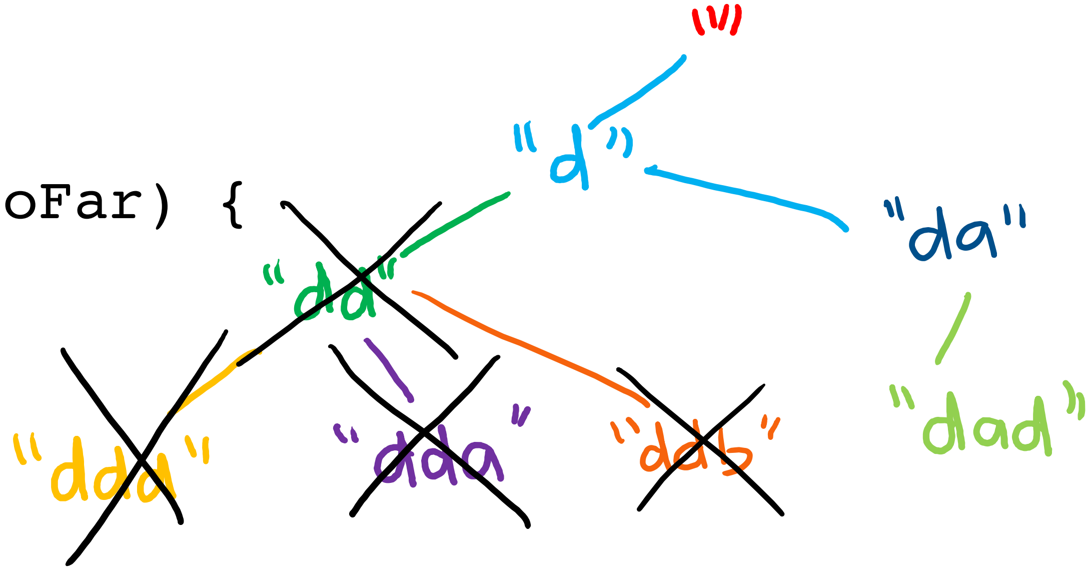
aaaa
aaab
aaba
aabb

Poll Everywhere

pollev.com/cse143

Suppose we had the following method:

```
public static void mystery(String soFar) {
     if (soFar.length() == 3) {
          System.out.println(soFar);
     } else {
          mystery(soFar + “d”);
          mystery(soFar + “a”);
          mystery(soFar + “b”);
     }
}
```



What is the **fourth** line of output of the call mystery(“”);

- This means you can stop once you’ve found 4 lines of output

output

ddd
dda
dd b
dad

diceRoll

Write a method `diceRoll` that accepts an integer parameter representing a number of 6-sided dice to roll, and output all possible arrangements of values that could appear on the dice.

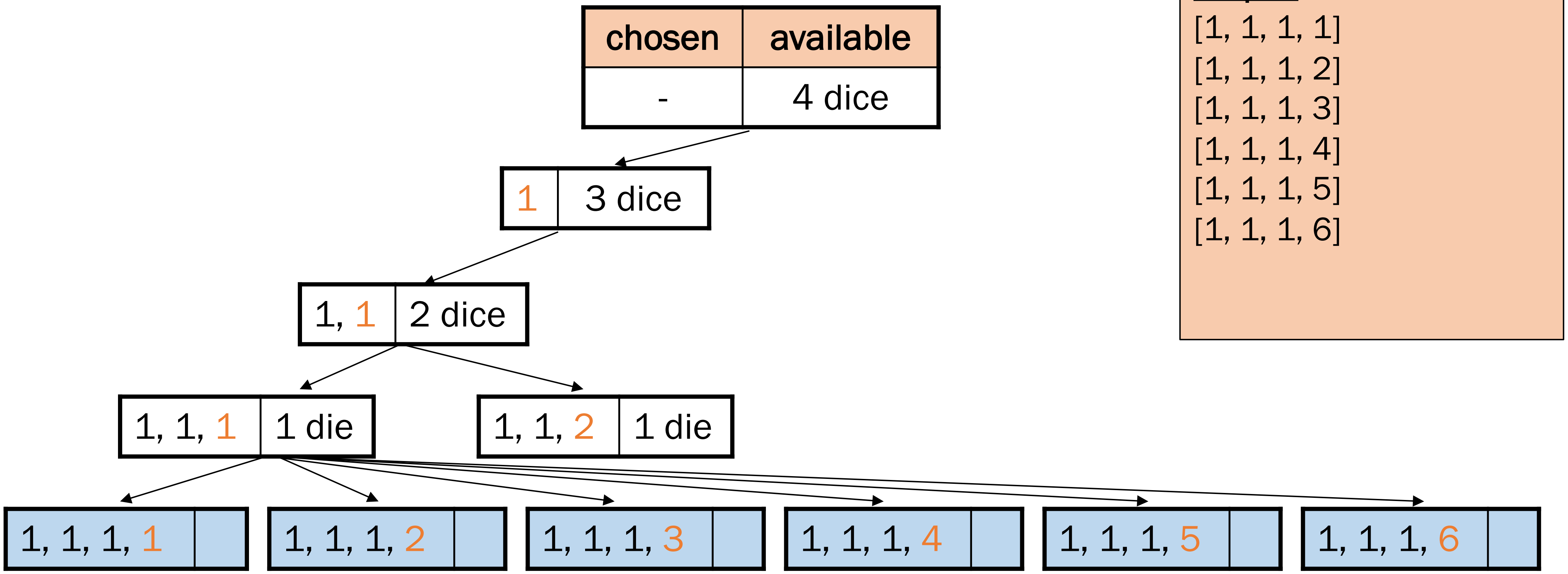
`diceRoll(2);`

```
[1, 1]  |  [3, 1]  |  [5, 1]
[1, 2]  |  [3, 2]  |  [5, 2]
[1, 3]  |  [3, 3]  |  [5, 3]
[1, 4]  |  [3, 4]  |  [5, 4]
[1, 5]  |  [3, 5]  |  [5, 5]
[1, 6]  |  [3, 6]  |  [5, 6]
[2, 1]  |  [4, 1]  |  [6, 1]
[2, 2]  |  [4, 2]  |  [6, 2]
[2, 3]  |  [4, 3]  |  [6, 3]
[2, 4]  |  [4, 4]  |  [6, 4]
[2, 5]  |  [4, 5]  |  [6, 5]
[2, 6]  |  [4, 6]  |  [6, 6]
```

`diceRoll(3);`

```
[1, 1, 1]
[1, 1, 2]
[1, 1, 3]
[1, 1, 4]
[1, 1, 5]
[1, 1, 6]
[1, 2, 1]
[1, 2, 2]
   ...
[6, 6, 4]
[6, 6, 5]
[6, 6, 6]
```

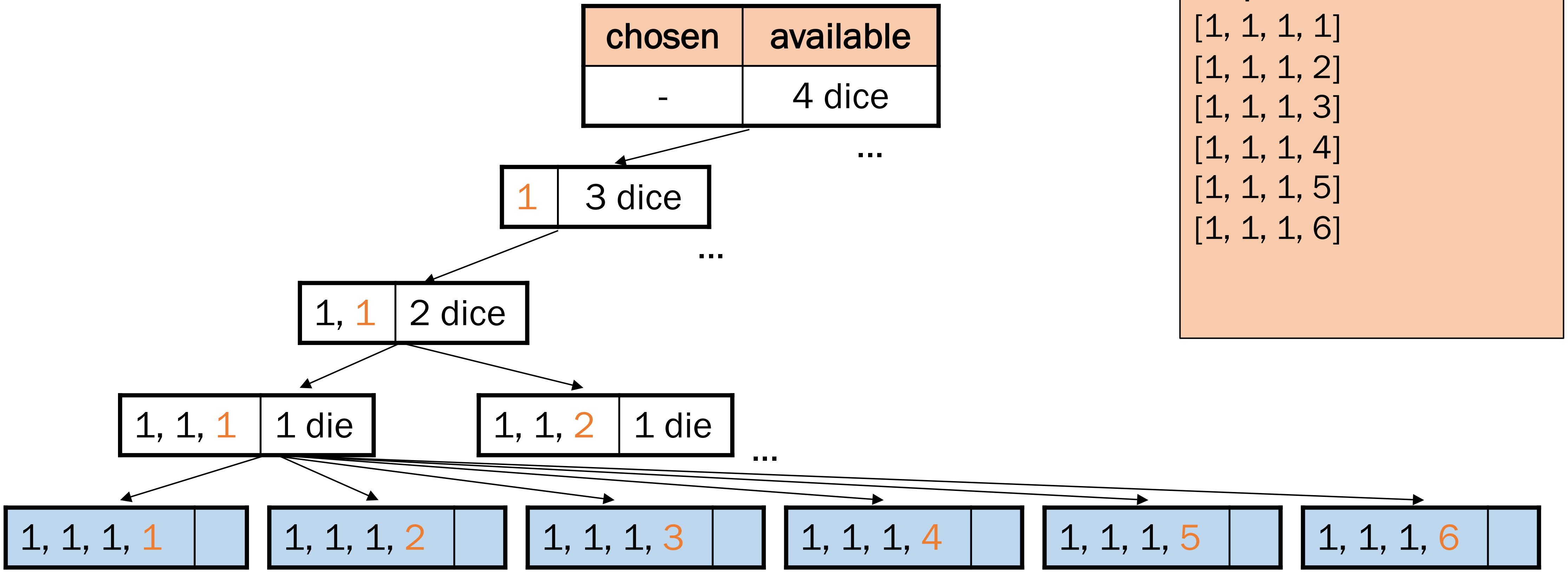
Decision Tree – diceRoll(4)



Output:

```
[1, 1, 1, 1]
[1, 1, 1, 2]
[1, 1, 1, 3]
[1, 1, 1, 4]
[1, 1, 1, 5]
[1, 1, 1, 6]
```

Decision Tree – diceRoll(4)



chosen	available
-	4 dice

...

1 | 3 dice

...

1, 1 | 2 dice

1, 1, 1 | 1 die

1, 1, 2 | 1 die

...

1, 1, 1, 1

1, 1, 1, 2

1, 1, 1, 3

1, 1, 1, 4

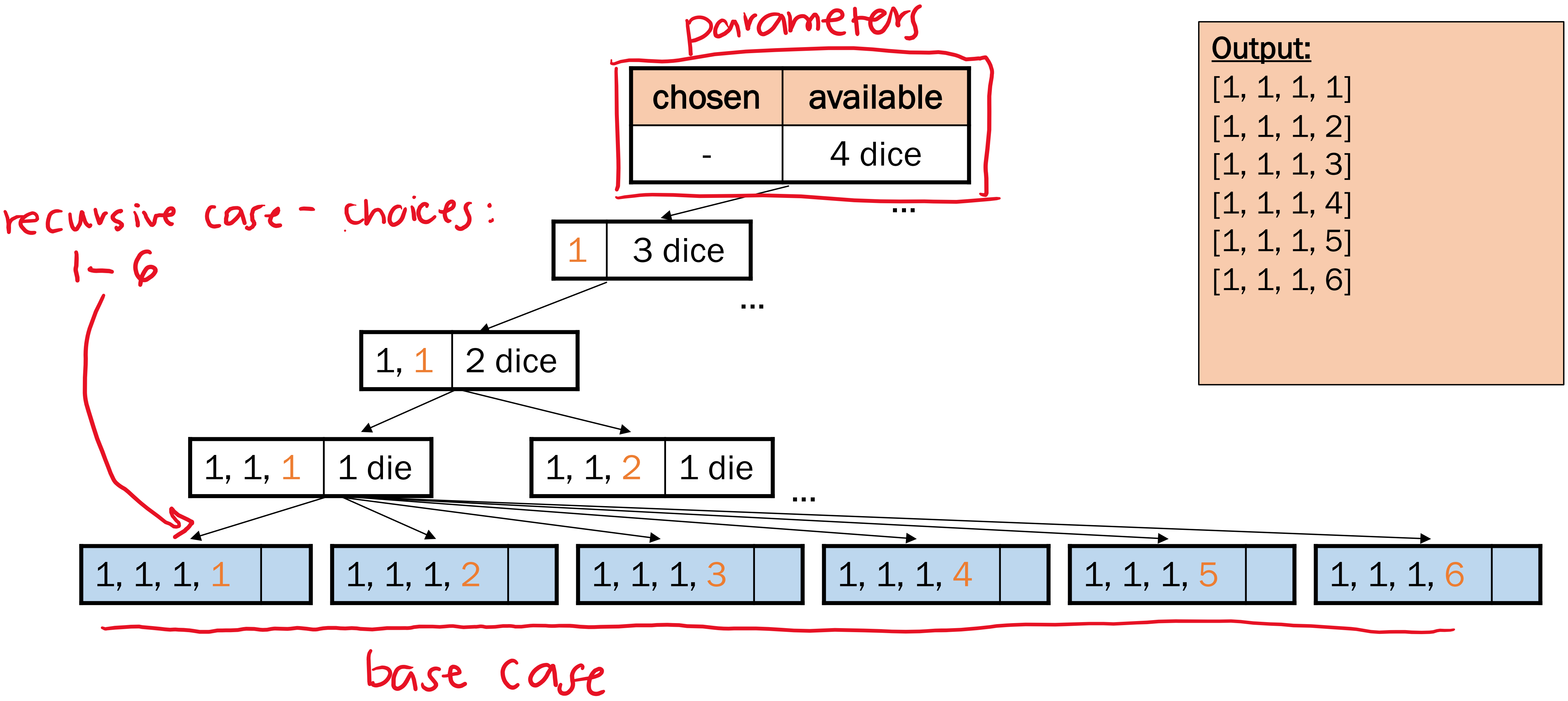
1, 1, 1, 5

1, 1, 1, 6

Output:

[1, 1, 1, 1]
[1, 1, 1, 2]
[1, 1, 1, 3]
[1, 1, 1, 4]
[1, 1, 1, 5]
[1, 1, 1, 6]

Decision Tree – diceRoll(4)



parameters

chosen	available
-	4 dice

recursive case - choices:
1 – 6

1	3 dice

1, 1	2 dice

1, 1, 1	1 die

1, 1, 2	1 die

base case

Output:
[1, 1, 1, 1]
[1, 1, 1, 2]
[1, 1, 1, 3]
[1, 1, 1, 4]
[1, 1, 1, 5]
[1, 1, 1, 6]