Probability theory and average-case complexity

Review of probability theory

Review of probability theory: **outcome**

- Examples:
  - Rolling a die and getting **1**
  - Rolling a die and getting **6**
  - Flipping three coins and getting **H, H, T**
  - Drawing two cards and getting 7 of hearts, 9 of clubs
- **NOT examples:**
  - Rolling a 6-sided die and getting an even number (this is **more than one outcome**—3 to be exact!)
  - Drawing a card and getting an ace (4 outcomes!)

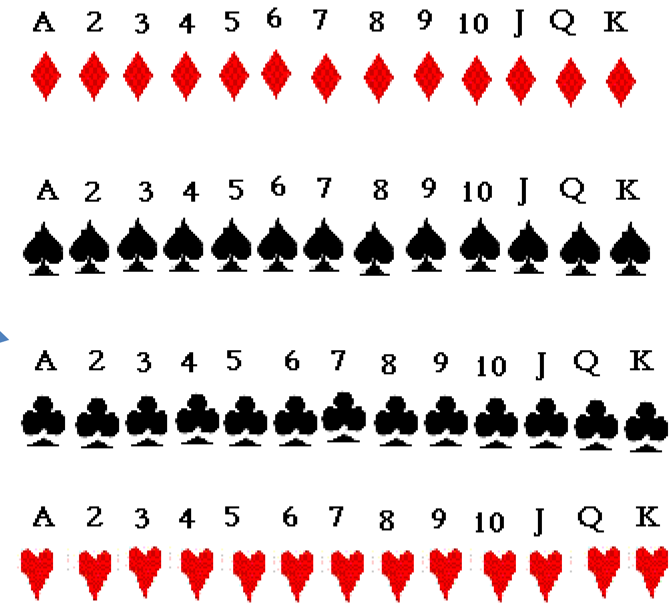
Review of probability theory: **event**

- Defn: **one or more** possible outcomes
- Examples:
  - Rolling a die and getting 1
  - Rolling a die and getting an even number
  - Flipping three coins and getting at least 2 “heads”
  - Drawing five cards and getting one of each suit

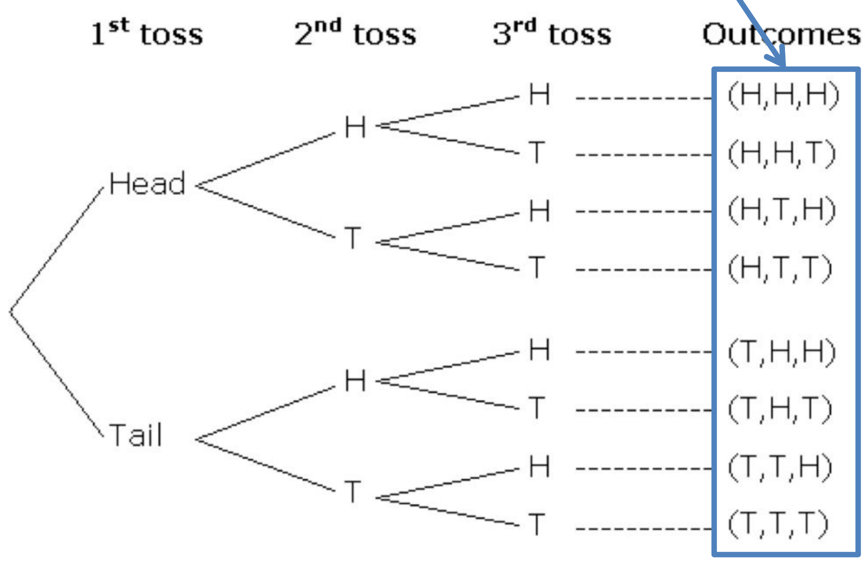
Review of probability theory: **sample space**

- Defn: A set of events
- Examples:

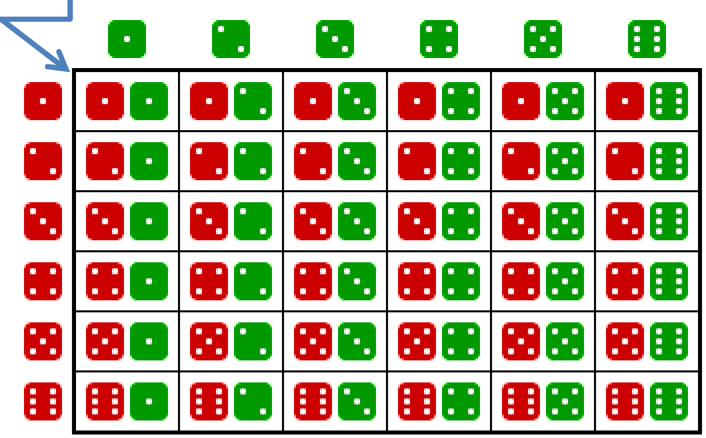
Drawing a card



Flipping 3 coins



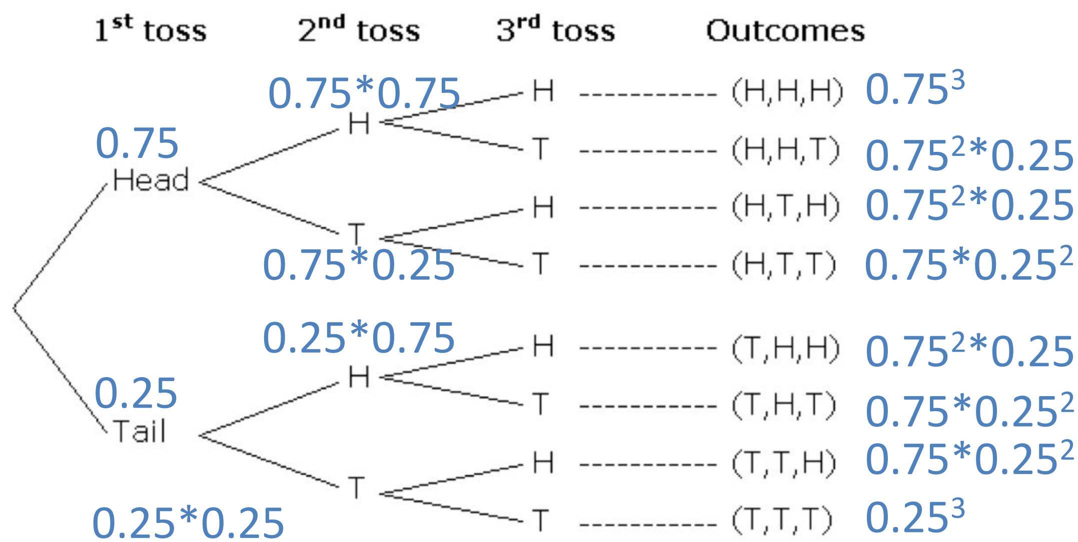
Rolling 2 dice



Review of probability theory: **probability distribution**

- **Idea: take a sample space S and add probabilities for each event**
- Defn: mapping from events of S to real numbers
  - $\Pr(A) \geq 0$ for any event A in S
  - $\sum_{A \in S} \Pr(A) = 1$
- Example:

Flipping 3 **biased** coins
75% heads, 25% tails



Review of probability theory: **probability of an event A**

- Defn: $\Pr(A) = \sum_{outcome \in A} \Pr(outcome)$
- Example:
  - Pr(roll a die and get even number) = Pr(roll a 2) + Pr(roll a 4) + Pr(roll a 6)

Review of probability theory: **random variable**

- **Idea: turn events into numbers**
- Let S be a sample space
- Defn: mapping from events to real numbers
- Example:

X = the number on a die after a roll
event “rolling a 1” -> 1
event “rolling a 2” -> 2
…
event “rolling a 6” -> 6

Technically, X is a function,
so we can write:
X(rolling a 1) = 1
X(rolling a 2) = 2
…
X(rolling a 6) = 6

Review of probability theory: **expected value of a random variable**

- **Idea: “average” value of the random variable**
- Remember: random variable **X** is a mapping from events in a sample space **S** to numbers
- Defn: Expected value of $\mathrm{X} = \mathrm{E[X]} =$

$$\sum_{event \in S} X(event) \Pr\big(X = X(event)\big)$$

- Short form: $\mathrm{E[X]} = \sum_x x \Pr(X = x)$ ← Here, x = X(event)
- Example: X = number on die after rolling

$$E[X] = \sum_{1 \le x \le 6} x \Pr(X = x) = 1\frac{1}{6} + 2\frac{1}{6} + \cdots + 6\frac{1}{6} = \frac{7}{2}$$

Expected running time of an algorithm

Expected running time of an algorithm

- Let A be an algorithm.
- Let S_n be the sample space of all inputs of size n.
- To talk about expected (/average) running time, we must specify how we **measure running time**.
  - We want to turn each input into a number (runtime).
  - **Random variables** do that…
- We must also specify how **likely** each input is.
  - We do this by specifying a **probability distribution over S_n.**

Expected running time of an algorithm

- Recall: algorithm A, sample space S_n
- We define a random variable
  ***$t_n(I)$* = number of steps taken by A on input I**
- We then obtain:

$$E(t_n) = \sum_{I \in S_n} t_n(I) \Pr(I)$$

- In this equation, I is an input in S_n, and
  Pr(I) is the probability of input I according to the probability distribution we defined over S_n
- **$E(t_n)$ is the average running time of A, given S_n**

Example time!

Example time: searching an array

- Let **L** be an array containing 8 distinct keys

```
Search(k, L[1..8]):
    for i = 1..8
        if L[i].key == k then return true
    return false
```

- **What should our sample space S_9 of inputs be?**
- Hard to reason about **all** possible inputs.
  - (In fact, there are uncountably infinitely many!)
- **Can group inputs by how many steps they take!**

Grouping inputs by how long they take

```
Search(k, L[1..8]):
    for i = 1..8
        if L[i].key == k then return true
    return false
```

- What causes us to return in loop iteration 1?
- How about iteration 2? 3? … 8? After loop?
- S_9 = { L[1]=k, L[2]=k, …, L[8]=k, k not in L }

- **Now we need a random variable for S_9!**

Using a random variable to capture running time

```
Search(k, L[1..8]):
    fo
                                  true
    ret
```

on takes 2 steps.

Do we have enough information to compute an answer?

- S_9 = { L[1]=k, L[2]=k, …, L[8]=k, k not in L }
- **Let T(e) = running time for event e in S_9**
- T(L[1]=k) = 2, T(L[2]=k) = 4, …, T(L[i]=k) = 2i
- T(k not in L) = 2*8+1 = 17
- **We then obtain:** $\boldsymbol{E[T] = \sum_{e \in S_9} T(e) Pr(e)}$

What about a probability distribution?

- We have a sample space and a random variable.
- **Now, we need a probability distribution.**
- **This is given to us in the problem statement.**
  - For each i, $\Pr(L[i] = k) = \frac{1}{16}$
  - $\Pr(k\ not\ in\ list) = \frac{1}{2}$

- If you don't get a probability distribution from the problem statement, you have to figure out how likely each input is, and come up with your own.

Computing the average running time

- **We now know:** $\boldsymbol{E[T] = \sum_{e \in S_9} T(e) Pr(e)}$
- T(e) = running time for event e in S_9
- T(L[i]=k) = 2i T(k not in L) = 17
- S_9 = { L[1]=k, L[2]=k, …, L[8]=k, k not in L}
- Probability distribution:
  - For each i, $\Pr(L[i] = k) = \frac{1}{16}$
  - $\Pr(k\ not\ in\ list) = \frac{1}{2}$
- **Therefore:** $\boldsymbol{E[T]} = T(L[1] = k)Pr(L[1] = k) + \cdots + T(L[8] = k)Pr(L[8] = k) + T(k\ not\ in\ L)Pr(k\ not\ in\ L)$

The final answer

- Recall: T(L[i]=k) = 2i T(k not in L) = 17
  - For each i, $\Pr(L[i] = k) = \frac{1}{16}$
  - $\Pr(k\ not\ in\ list) = \frac{1}{2}$
- $\boldsymbol{E}[\boldsymbol{T}] = T(L[1] = k)Pr(L[1] = k) + \cdots + T(L[8] = k)Pr(L[8] = k) + T(k\ not\ in\ L)Pr(k\ not\ in\ L) = \frac{2}{16} + \frac{4}{16} + \cdots + \frac{16}{16} + \frac{17}{2} = 13$
- **Thus, the average running time is 13.**

Slightly harder problem: L[1..n]

```
Search(k, L[1..n]):
    for i = 1..n
        if L[i].key == k then return true
    return false
```

- **Problem:** what is the average running time of Search, given the following probabilities?
  - $\Pr(L[i] = k) = \frac{1}{2n}$
  - $\Pr(k\ not\ in\ L) = \frac{1}{2}$

Computing *E[T]:* part 1

```
Search(k, L[1..n]):
    for i = 1..n
        if L[i].key == k then return true
    return false
```

- What is our sample space?
  - S_{n+1} = { L[1]=k, L[2]=k, …, L[n]=k, k not in L }
- What is our random variable?
  - Let T(e) = running time for event e in S_{n+1}
- What is the running time of each event?
  - T(L[i]=k) = 2i, T(k not in L)=2n+1

Computing ***E[T]:*** part 2

- What we know:
  - S_{n+1} = { L[1]=k, L[2]=k, …, L[n]=k, k not in L }
  - T(L[i]=k) = 2i, T(k not in L)=2n+1
  - $\Pr(k\ not\ in\ L) = \frac{1}{2}$ and $\Pr(L[i] = k) = \frac{1}{2n}$
- Now we can compute $\boldsymbol{E}[\boldsymbol{T}] = \sum_{\boldsymbol{e} \in \boldsymbol{S}_{\boldsymbol{n}+\mathbf{1}}} \boldsymbol{T}(\boldsymbol{e})\boldsymbol{Pr}(\boldsymbol{e})$.
  - $\boldsymbol{E}[\boldsymbol{T}] =$
    $\mathrm{T}(\mathrm{L}[1] = \mathrm{k})\,\Pr(\mathrm{L}[1] = \mathrm{k}) +$
    $\mathrm{T}(\mathrm{L}[2] = \mathrm{k})\,\Pr(\mathrm{L}[2] = \mathrm{k}) + \cdots +$
    $\mathrm{T}(\mathrm{L}[\mathrm{n}] = \mathrm{k})\,\Pr(\mathrm{L}[\mathrm{n}] = \mathrm{k}) +$
    $\mathrm{T}(k\ not\ in\ L)\,\Pr(k\ not\ in\ L)$
  - $\boldsymbol{E}[\boldsymbol{T}] = 2\frac{1}{2n} + 4\frac{1}{2\mathrm{n}} + \cdots + 2\mathrm{n}\frac{1}{2\mathrm{n}} + (2\mathrm{n} + 1)\frac{1}{2}$

Computing *E[T]:* part 3

- $\boldsymbol{E}[\boldsymbol{T}] = 2\frac{1}{2n} + 4\frac{1}{2\mathrm{n}} + \cdots + 2\mathrm{n}\frac{1}{2\mathrm{n}} + (2\mathrm{n}+1)\frac{1}{2}$
- $\boldsymbol{E}[\boldsymbol{T}] = \frac{1}{n}(1+2+\cdots+n) + (2n+1)\frac{1}{2}$
- $\boldsymbol{E}[\boldsymbol{T}] = \frac{1}{n}\frac{n(n+1)}{2} + \frac{2n+1}{2} = \frac{n+1}{2} + \frac{2n+1}{2} = \frac{3n}{2} + 1$
- **Thus, Search(k, L[1..n]) has expected (or average) running time 3n/2+1 for the given probabilities.**