CS630 Graduate Algorithms

October 17, 2024

by Dora Erdos and Jeffrey Considine

- Randomized algorithms
  - review xxx
- Contention Resolution KT ch. 13.1

Why randomized algorithms?

deterministic algorithms to solve problems:

- find an exact (or optimal) solution
- worst-case running time

approximation algorithms:

- solution is acceptable close to optimal
- worst case running time is better than for the optimal algorithm
    - but may still be infeasible

randomization:

- deterministic algorithms on random input
    - look at average case running time, e.g. randomized quicksort
- algorithms that behave random
    - they are “good” if they succeed with some probability
    - “succeed” - either in terms of runtime or quality of solution

using randomness to get definitive results - example

2 biased coins, head is 60% likely. How can we use this to simulate an unbiased coin?

$\frac{6}{10} = \frac{3}{5}$

flip twice:

ignore and do again → HH $= 0.6^2 = \frac{9}{25}$

heads → HT $= \frac{3}{5} \times \frac{2}{5} = \frac{6}{25}$

tails → TH $= \frac{6}{25}$

ignore and repeat → TT $= \frac{2}{5} \times \frac{2}{5} = \frac{4}{25}$

using randomness to get definitive results - example 2

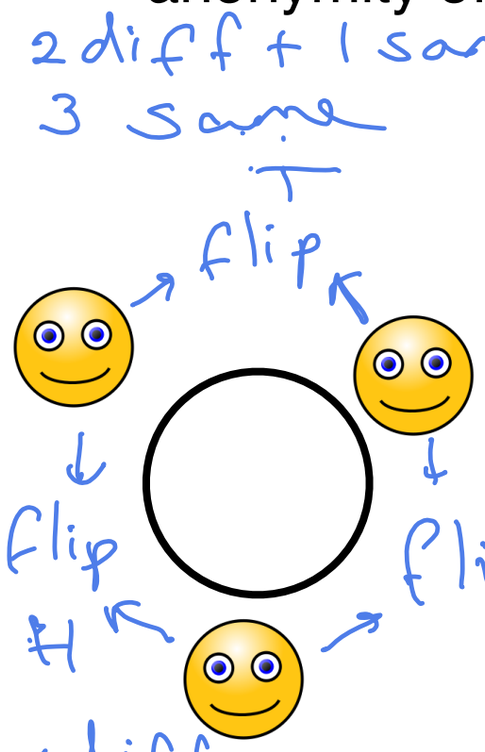
conclusion:
even # diff ⇒ nobody paid
odd # diff ⇒ one friend paid

3 friends are sitting at a round table at the restaurant. At the end of the meal the restaurant owner tells them that the meal has already been paid for.

The owner doesn't reveal who has paid. But tells them that either (1.) one of the friends has secretly paid in advance (2.) the meal was on the house (= free)

nobody lies

Design a strategy to decide whether one of them has paid, while maintaining the anonymity of the person.



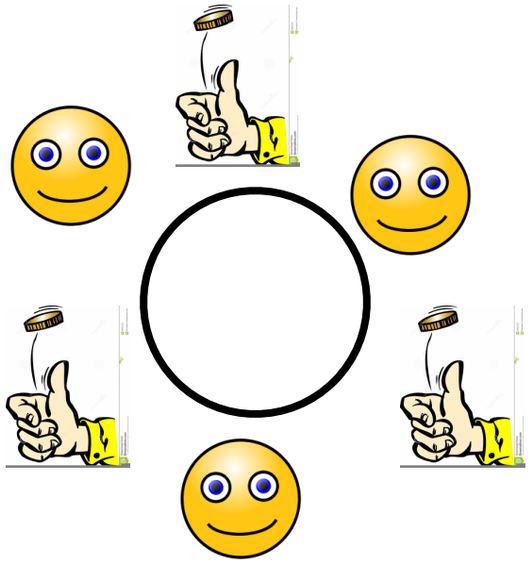
- each person sees the outcome of 2 of the three coinflips
  → they know whether the two coins are the same or different.
- tell the other whether same or not
  → tell the truth → didn't pay
  → lie → person who paid

using randomness to get definitive results - example 2

Algorithm:

1. every pair of people flip a coin only visible to the pair (that is, 3 flips total)
2. each person reports whether they have seen an even or odd number of heads
   - the person who paid (if there is such) will lie
   - the other people will be truthful

claim: if the sum of the claims is even, then nobody lied, if the sum of claims is odd then one person lied.



probability exercises

propapility of genders for 2 children is independent

A family has two children.

- What is the probability that both are girls?

$\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$

- When asked, the father says that his older child is a girl. Given this information, what is the probability of having two daughters?

$\frac{1}{2}$

- When asked, the father says that he has a daughter. Given this information, what is the probability of having two girls?

GG : $\frac{1}{4}$

GB = $\frac{1}{4}$ } $\frac{3}{4}$

BG = $\frac{1}{4}$

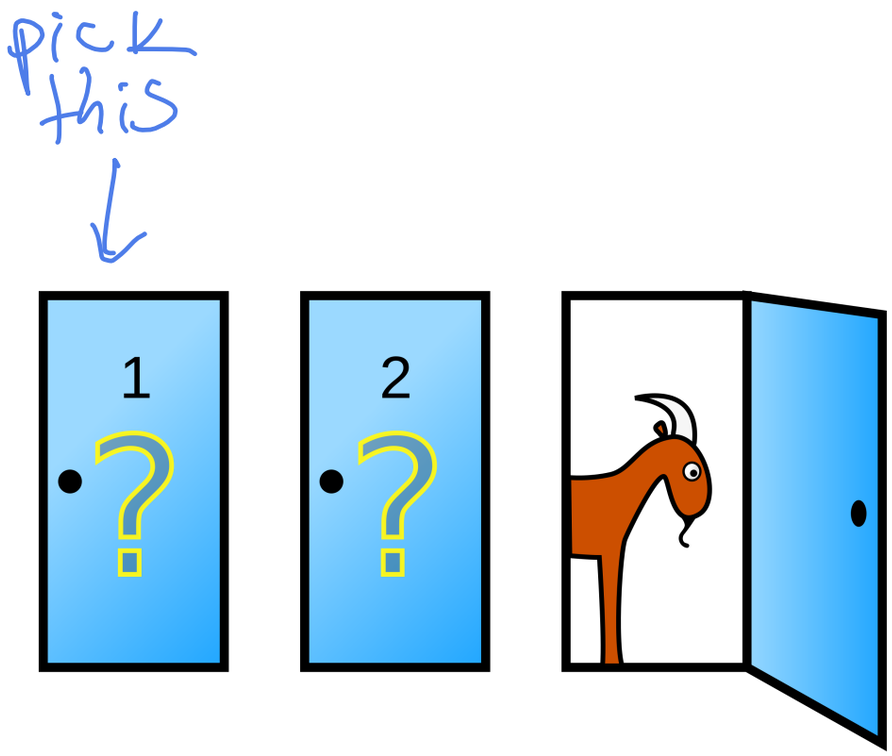
probability exercises

Monty Hall problem:

(Monty Hall was the host of the program Let's Make a Deal)

There are 3 doors, behind one of the doors is a car, the other two doors hide goats.

- The contestant selects one of the doors
- The host then opens one of the remaining two doors that hides a goat
- After this, should the contestant change their selection?



Monty Hall - TopHat

The contestant picks door 1. Then Monty reveals that door 3 hides a goat. What is the probability that door 2 hides the car? (Thus, should the contestant switch to door 2?)

A. P(car behind door 2) = 1/4

B. P(car behind door 2) = 1/3

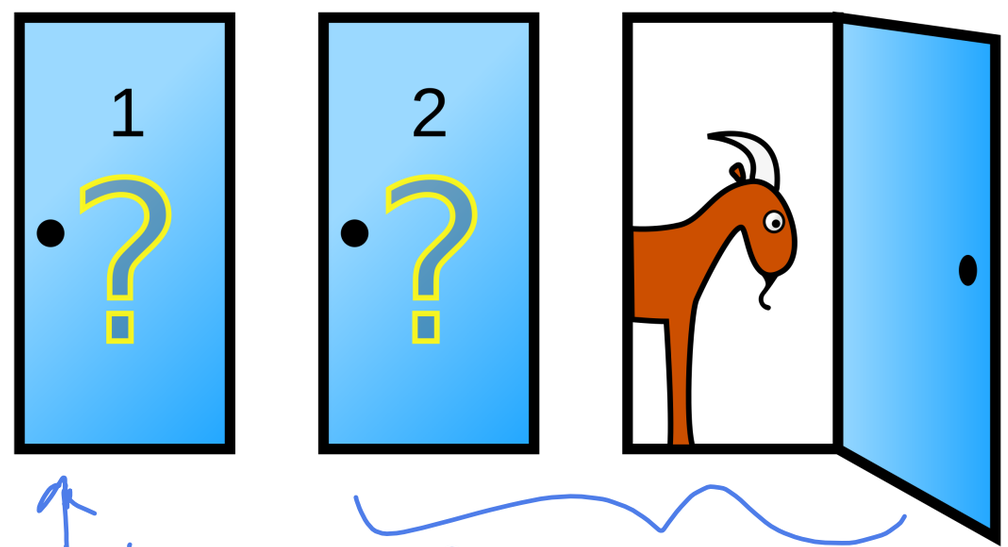
C. P(car behind door 2) = 1/2

D. P(car behind door 2) = 2/3

door 1	door 2	door 3	stay	move
car	goat	goat	car	goat
goat	car	goat	goat	car
goat	goat	car	goat	car.

$\frac{1}{3}$ we get car

initial $p(\text{car 1}) = \frac{1}{3}$ $p(\text{car in 2 or 3}) = \frac{2}{3}$



after: still $\frac{1}{3}$ $\frac{2}{3}$, but door 3 has a goat

Conditional probabilities

Ω is the space of all events, A and B be are two events.

conditional probability: occurrences of A that overlap with B

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

normalizing factor: $\frac{P(\Omega \mid B)}{P(B)} = 1$

Bayes' theorem: $P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$

Law of total probability: $P(A) = P(A \mid B)P(B) + P(A \mid \overline{B})P(\overline{B})$

Chain rule:

$$P(A_n \cap A_{n-1} \cap \ldots A_1) = \prod_{k=1}^{n} P\left(A_k \mid \cap_{j=1}^{k-1} A_j\right)$$

$$P(A_1 \cap A_2 \cap A_3 \cap A_4) = P(A_4 \mid A_3 \cap A_2 \cap A_1) \cdot P(A_3 \mid A_2 \cap A_1) \cdot P(A_2 \mid A_1) \cdot P(A_1)$$

See final slides in deck for explanation!

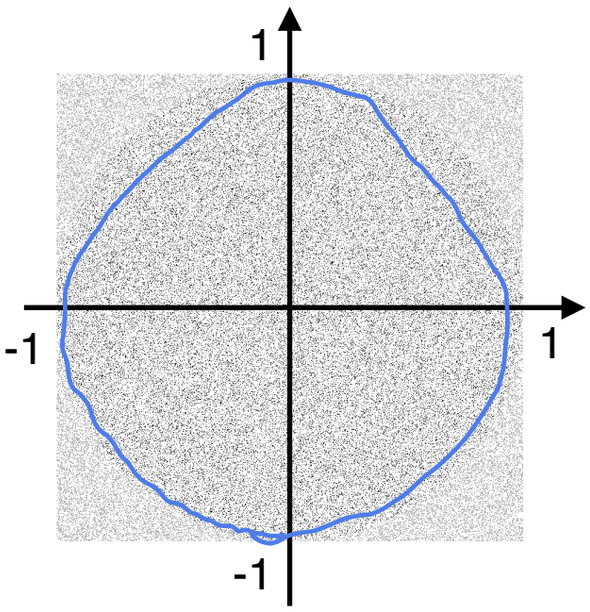
Random running time vs. random result - Las Vegas

binary array A of length n with n/2 0s and 1s. Find an index in A containing 1.

```
//Las Vegas algorithm
input: A
repeat:
        k= randint(n)
        if A[k] == 1:
            return k
```

Compute a random point in the unit circle

```
repeat:
   x = rand(-1:1)
   y = rand(-1:1)
   if x²+y²<1:
       return (x,y)
```



pros:
- simple
- output is guaranteed to be correct.

cons:
- runtime can be very long (even infinite)

circle example: https://yourbasic.org/algorithms/las-vegas/

Random running time vs. random result - Monte Carlo

binary array A of length n with n/2 0s and 1s. Find an index in A containing 1.

```
//Monte Carlo algorithm
input: A, k
repeat k times:
        r = randint(n)
        if A[r] == 1:
                return r
return r
```

pros:

- bounded runtime

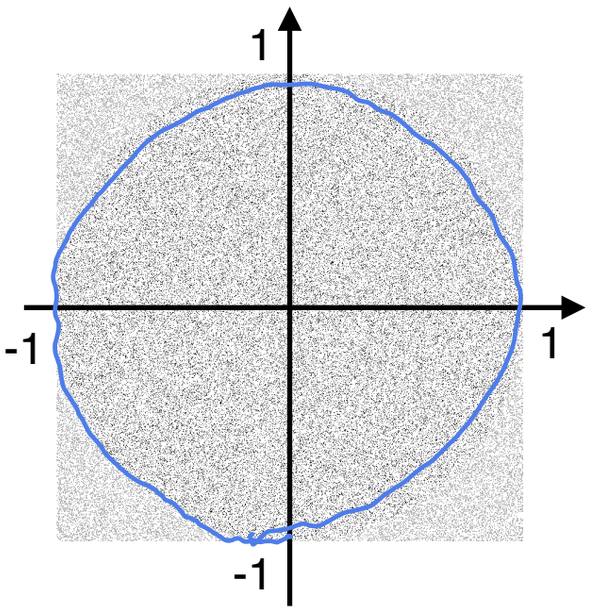
cons:

- correct only a percentage of the time.

Estimate the value of π

ratio of points randomly falling in the circle is $\approx \pi/4$

```
input : n = 10000
count = 0
repeat n times:
  x = rand(-1:1)
  y = rand(-1,1)
  if x²+y²<1:
      count++
return 4*count/n
```



circle example: https://yourbasic.org/algorithms/las-vegas/

Random running time vs. random result

Las Vegas method: repeat random trial/process until success

+ guaranteed correct result

- (expected) number of trials might be large

need: find expected runtime

Monte Carlo method: repeat random trial a fixed number of times

+ finds the correct solution with some probability

- deterministic running time

need: decide on number of trials, such that the probability of success is high

Contention resolution

n processes P_1, P_2, …, P_n are competing for access to a single shared database

- time is divided into discrete rounds
- at most one process can access the database in a single round
- if more than one processes try to access it, they are all locked out for the round

How can we divide the rounds so that all processes can get through on a regular basis?

if they can communicate: agree on an order

they can't communicate:

in each round each process P_i decides at random with some prob. p to make an attempt.

Contention resolution

n processes P_1, P_2, …, P_n are competing for access to a single shared database

- time is divided into discrete rounds
- at most one process can access the database in a single round
- if more than one processes try to access it, they are all locked out for the round

How can we divide the rounds so that all processes can get through on a regular basis?

- if the processes can communicate — agree on some order of access

- suppose they can't:
  - in each round each process makes an attempt with probability p
  - what is a good value for p? $\frac{1}{n}$ might be good
  - how good is this strategy?

Measuring performance

In each round each process makes an attempt with probability p

How should we measure the performance of this algorithm?

- min the probability that more than one try to access
- min the # of rounds that there is a clash
- min the # of rounds until every process gets through
  - ↳ select a p suitable for this
  - ↳ objective function: expected # of rounds

Measuring performance

In each round each process makes an attempt with probability p

goal: compute the expected number of rounds until every process had access

events:

$A[i,t]$ = the event that process P_i makes an attempt at time t

$S[i,t]$ = the event that P_i makes an attempt at time t and succeeds

$P(A[i,t]) = p$

Measuring performance - TopHat

In each round each process makes an attempt with probability p

events:

$A[i,t]$ = the event that process P_i makes an attempt at time t

$S[i,t]$ = the event that P_i makes an attempt at time t and succeeds

What is the probability that Pi makes an attempt at time t and succeeds?

A. $P(S[i,t]) = p$

B. $P(S[i,t]) = 1 - p$

C. $P(S[i,t]) = p^n$

D. $P(S[i,t]) = p(1-p)^{n-1}$

Measuring performance

In each round each process makes an attempt with probability p

$A[i,t]$ = the event that process P_i makes an attempt at time t

$S[i,t]$ = the event that P_i makes an attempt at time t and succeeds

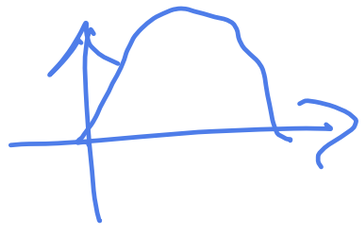
probability of success (given n processes):

$p(S[i,t]) =$ P_i makes attempt, but the others don't

$= p(1-p)^{n-1}$

are these probabilities independent?

yes → by design of algo



What value of p can maximize this probability?

$f(p) = p(1-p)^{n-1}$ $\quad f(0)=0 \quad f(1)=0 \quad f(p)\geq 0$

$f'(p) = (1-p)^{n-1} - p(n-1)(1-p)^{n-2}$ ⟹ solve this

$(g\cdot h)' = g'h - gh'$

$f'(p) = 0$

$p = \frac{1}{n}$

Measuring performance

In each round each process makes an attempt with probability p

A[i,t] = the event that process P_i makes an attempt at time t

S[i,t] = the event that P_i makes an attempt at time t and succeeds

probability of success (given n processes):

$$P(S[i,t]) = p(1-p)^{n-1}$$

are these probabilities independent?

- yes, that is how the algorithm is designed

What value of p can maximize this probability?

$$f(p) = p(1-p)^{n-1}$$

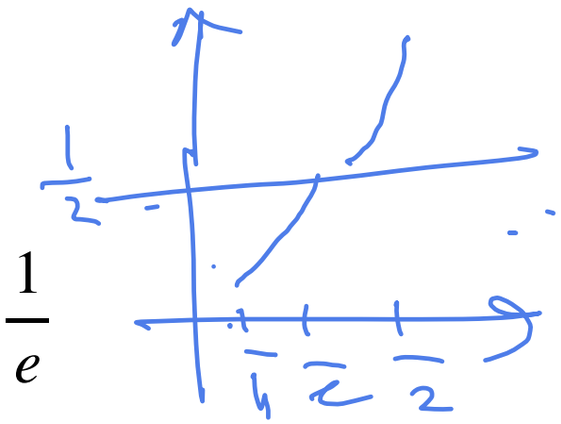
- f(p) is a nonnegative function, with f(0) = f(1) = 0.
- take the derivative to find its max $f'(p) = (1-p)^{n-1} - (n-1)p(1-p)^{n-2}$
- $f'(p)$ is maximum when $p = 1/n$

Expected success at time t

probability of Pi succeeding at time t: $P(S[i,t]) = p(1-p)^{n-1}$

for $p = 1/n$ we get $P(S[i,t]) = \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1}$

- value converges up from 1/4 up to 1/e $\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n} = \frac{1}{e}$
- value converges down from 1/2 down to 1/e $\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n-1} = \frac{1}{e}$



lower bound

We get $\frac{1}{en} \le P(S[i,t]) \le \frac{1}{2n}$ hence $P(S[i,t]) = \Theta\left(\frac{1}{n}\right)$

Expected wait time for success

probability of P_i succeeding at time t with p=1/n: $P(S[i,t]) = \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1}$

we have: $P(S[i,t]) = \Theta\left(\frac{1}{n}\right)$

How long do we have to wait for success?

$F([i,t])$ = the event that P_i doesn't succeed *after* t rounds

The probability that we don't succeed

- after O(t) rounds
  - previous slide for each t we have $\frac{1}{en} \leq P(S[i,t])$
  - $F([i,t]) = \Pi_{r=1}^{t}\overline{P(S[i,t])} \leq \left(1-\frac{1}{en}\right)^{t}$
  - after $t = \lceil en \rceil$ we get $F([i,t]) \leq \frac{1}{e}$ ← after $\Theta(n)$ rounds this bound is a const.

Expected wait time for success

probability of P_i succeeding at time t with p=1/n: $P(S[i,t]) = \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1}$

we have: $P(S[i,t]) = \Theta\left(\frac{1}{n}\right)$

How long do we have to wait for success?

conclusion:
after $O(n \ln n)$ rounds (huge!)
the probability of every process getting access is almost 1

$F([i,t])$ = the event that P_i doesn't succeed *after* t rounds

-1

$\frac{1}{e} = e$

The probability that we don't succeed

- after $\lceil en \rceil$ (thus O(n)) rounds we have $P(F[i,t]) \le e^{-1}$

can we get the prob close to 0?

large enough constant (chosen by the user) (c comes from the definition of Θ)

- after $t = c\lceil en \rceil n \ln n$ rounds we get $t = \Theta(\lceil en \rceil n \ln n)$
-
  - $\left(\left(1-\frac{1}{en}\right)^{\lceil en \rceil}\right)^{c \ln n} = \left(e^{-1}\right)^{c \ln n} = n^{-c} = \frac{1}{n^c}$

substitute t is $\left(1-\frac{1}{en}\right)^t$

Expected wait time for all processes to succeed

probability of P_i succeeding at time t with p=1/n: $P(S[i,t]) = \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} = \Theta\left(\frac{1}{n}\right)$

How many rounds before *all* processes get through?

F([i,t]) = the event that P_i doesn't succeed after t rounds

probability that t rounds are not enough: $P\left(\cup_{i=1}^{n} F[i,t]\right)$

Union Bound: given events $E_1, E_2, .., E_n$ we have

$$P\left(\cup_{i=1}^{n} E_i\right) \le \sum_{i=1}^{n} P(E_i)$$

We know $P(F[i,t]) \le n^{-c}$ → prob. of P_i failing is low ⇒ we need every process

so by the union bound $P\left(\cup_{i=1}^{n} F[i,t]\right) \le \sum_{i=1}^{n} P(F[i,t]) \le n \cdot n^{-c} = \frac{1}{n^{c-1}}$

If we select $t = 2\lceil en \rceil \ln n$ then this result implies we succeed with prob at least $1 - n^{-1}$

Conditional probabilities

Ω is the space of all events, A and B be are two events.

conditional probability: occurrences of A that overlap with B

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

normalizing factor: $\frac{P(\Omega \mid B)}{P(B)} = 1$

Bayes' theorem: $P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$

two events are independent if $P(A \mid B) = P(A) \Rightarrow$ we can't predict anything about A by knowing something about B

- events A_1, A_2,… are independent if we can't say anything about A_j if we have information about any (or a combination) of the other A_is
- events are pairwise independent if A_i and A_j are independent for any pair i,j
  - pairwise vs total independence: A_1 = coin 1 turns up heads, A_2 = coin 2 turns up heads, A_3 = coins 1 and 2 are the same

Conditional probabilities

Ω is the space of all events, A and B be are two events.

conditional probability: occurrences of A that overlap with B

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

normalizing factor: $\frac{P(\Omega \mid B)}{P(B)} = 1$

Bayes' theorem: $P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$

Law of total probability: $P(A) = P(A \mid B)P(B) + P(A \mid \overline{B})P(\overline{B})$

example:

$P(class \mid snow) = 3/5$

$P(class \mid no\ snow) = 99/100$

$P(snow) = 1/10$

$$\Rightarrow P(class) = 3/5 \cdot 1/10 + 99/100 \cdot (1 - 1/10) = 0.951$$

Conditional probabilities

Ω is the space of all events, A and B be are two events.

conditional probability: occurrences of A that overlap with B

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

normalizing factor: $\frac{P(\Omega \mid B)}{P(B)} = 1$

Bayes' theorem: $P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$

Law of total probability: $P(A) = P(A \mid B)P(B) + P(A \mid \overline{B})P(\overline{B})$

Chain rule:

$$P(A_n \cap A_{n-1} \cap \ldots A_1) = \prod_{k=1}^{n} P\left(A_k \mid \cap_{j=1}^{k-1} A_j\right)$$

$$P(A_1 \cap A_2 \cap A_3 \cap A_4) = P(A_4 \mid A_3 \cap A_2 \cap A_1) \cdot P(A_3 \mid A_2 \cap A_1) \cdot P(A_2 \mid A_1) \cdot P(A_1)$$