Gonit Adda

Find a prime number in $[2^k, 2^{k+1}]$

Find a prime number in $[2^k, 2^{k+1}]$

1. **Bertrand's postulate** tells us that such a prime exists.

Find a prime number in $[2^k, 2^{k+1}]$

1. **Bertrand's postulate** tells us that such a prime exists.
2. **Brute force** tells us that we can find such a prime deterministically.

Find a prime number in $[2^k, 2^{k+1}]$

1. **Bertrand's postulate** tells us that such a prime exists.

2. **Brute force** tells us that we can find such a prime deterministically.

But if we didn't have to stick to doing this problem deterministically, can we do better?

Tangents

Given the faces of a convex polyhedron in 3D, find its volume.

Tangents

Given the faces of a convex polyhedron in 3D, find its volume.

Turns out, doing this in polynomial time is really hard!

Tangents

Given the faces of a convex polyhedron in 3D, find its volume.

Turns out, doing this in polynomial time is really hard!

But take a bounding box. Throw darts inside the bounding box randomly.

Tangents

Given the faces of a convex polyhedron in 3D, find its volume.

Turns out, doing this in polynomial time is really hard!

But take a bounding box. Throw darts inside the bounding box randomly.

Take fraction of darts that land inside the convex polyhedron. This can be checked quickly!

Tangents

Given the faces of a convex polyhedron in 3D, find its volume.

Turns out, doing this in polynomial time is really hard!

But take a bounding box. Throw darts inside the bounding box randomly.

Take fraction of darts that land inside the convex polyhedron. This can be checked quickly!

Assuming a convex polyhedron with counterclockwise oriented faces (seen from outside), it is sufficient to check the point is behind all faces. Take the vector from the point to each face and check the sign of the scalar product with the face's normal. If it is positive, the point is behind the face; if it is zero, the point is on the face; if it is negative, the point is in front of the face. (**source - StackOverflow**)

So adding randomness to our algorithm gives us additional power.

So adding randomness to our algorithm gives us additional power.

Complexity is the study of how limiting or adding resources aids/harms computation.

So adding randomness to our algorithm gives us additional power.

Complexity is the study of how limiting or adding resources aids/harms computation.

Randomness is a very powerful resource...or is it?

So adding randomness to our algorithm gives us additional power.

Complexity is the study of how limiting or adding resources aids/harms computation.

Randomness is a very powerful resource...or is it?

Find a max-cut of a graph.

So adding randomness to our algorithm gives us additional power.

Complexity is the study of how limiting or adding resources aids/harms computation.

Randomness is a very powerful resource...or is it?

Find a max-cut of a graph.

This is an NP hard problem! So no deterministic solution!

This is an NP hard problem! So no deterministic solution!

But maybe we could just be happy with an approximate solution?

This is an NP hard problem! So no deterministic solution!

But maybe we could just be happy with an approximate solution?

> Approximate a max-cut of a graph upto a factor of c.

This is an NP hard problem! So no deterministic solution!

But maybe we could just be happy with an approximate solution?

> Approximate a max-cut of a graph upto a factor of c.

We will show that we can do $c = 2$ (or $1/2$,, depending on how you define your constants).

This is an NP hard problem! So no deterministic solution!

But maybe we could just be happy with an approximate solution?

> Approximate a max-cut of a graph upto a factor of c.

We will show that we can do $c = 2$ (or $1/2$,, depending on how you define your constants).

So we apply the probabilistic method.

Consider a cut uniformly at random.

Let $C =$ number of crossing edges.

This is an NP hard problem! So no deterministic solution!

But maybe we could just be happy with an approximate solution?

Approximate a max-cut of a graph upto a factor of c.

We will show that we can do $c = 2$ (or $1/2$,, depending on how you define your constants).

So we apply the probabilistic method.

Consider a cut uniformly at random.

Let $C =$ number of crossing edges.

For each edge e in our graph, what is the probability that e is a crossing edge?

For each edge e in our graph, what is the probability that e is a crossing edge?

Suppose $e = uv$.

For each edge e in our graph, what is the probability that e is a crossing edge?

Suppose $e = uv$.

Then we want u and v to be in different sets. Suppose the two sets are A and B. Then

$$P(\text{u and v are in different sets}) = P(\text{u in A, v in B}) + P(\text{u in B, v in A})$$

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

For each edge e in our graph, what is the probability that e is a crossing edge?

Suppose $e = uv$.

Then we want u and v to be in different sets. Suppose the two sets are A and B. Then

$$P(\text{u and v are in different sets}) = P(\text{u in A, v in B}) + P(\text{u in B, v in A})$$

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

Now $C = \sum_{\text{e is an edge}} I_e$ where $I_e = 1$ if it is a crossing edge and 0 otherwise.

For each edge e in our graph, what is the probability that e is a crossing edge?

Suppose $e = uv$.

Then we want u and v to be in different sets. Suppose the two sets are A and B. Then

$$P(\text{u and v are in different sets}) = P(\text{u in A, v in B}) + P(\text{u in B, v in A})$$

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

Now $C = \sum_{\text{e is an edge}} I_e$ where $I_e = 1$ if it is a crossing edge and 0 otherwise.

Now applying linearity, we get

$$E[C] = \frac{E}{2}$$

and so some cut exists that achieves a good enough approximation.

Now applying linearity, we get

$$E[C] = \frac{E}{2}$$

and so some cut exists that achieves a good enough approximation.

But this requires us to go over every single cut, if we could do that, we might as well just find the max cut.

How do we derandomize an algorithm?

Method 1. Pick an vertex, and add it to the partition that has more edges incident with our vertex.

Method 1. Pick an vertex, and add it to the partition that has more edges incident with our vertex.

This achieves $\frac{E}{2}$. Why?

Method 1. Pick an vertex, and add it to the partition that has more edges incident with our vertex.

This achieves $\frac{E}{2}$. Why?

Method 2. Pairwise Independent Hash functions. This is really nice, but maybe some other day.

Method 2. Pairwise Independent Hash functions. This is really nice, but maybe some other day.

But the idea is that for the probabilistic method, we go over every cut. But what if could pick only a few cuts and one of them was guaranteed to have $\frac{E}{2}$ crossing edges?

Method 2. Pairwise Independent Hash functions. This is really nice, but maybe some other day.

But the idea is that for the probabilistic method, we go over every cut. But what if could pick only a few cuts and one of them was guaranteed to have $\frac{E}{2}$ crossing edges?

Turns out you can pick n cuts that do this. Hint: binary.

Takeaway: There are ways to derandomize algorithms. In fact, we can almost certainly derandomize an algorithm that takes polynomial time! This conjecture is known as P = BPP. So adding randomness to a task that takes polynomial time does not contribute anything!

Back To Primes

Find a prime number in $[2^k, 2^{k+1}]$

Back To Primes

Find a prime number in $[2^k, 2^{k+1}]$

Let us find a randomized algorithm.

Back To Primes

Find a prime number in $[2^k, 2^{k+1}]$

Let us find a randomized algorithm.

The algorithm is simply picking an arbitrary number and checking if it is prime.

Back To Primes

Find a prime number in $[2^k, 2^{k+1}]$

Let us find a randomized algorithm.

The algorithm is simply picking an arbitrary number and checking if it is prime.

This is very efficient and will almost sure work in polynomial time, but how to prove?

By the prime number theorem, there are roughly $\frac{N}{\log N}$ primes in $[1, N]$.

By the prime number theorem, there are roughly $\frac{N}{\log N}$ primes in $[1, N]$.

So there are about $\frac{2^k}{k}$ primes in $[1, 2^k]$, and $\frac{2^{k+1}}{k+1}$ in $[1, 2^{k+1}]$

By the prime number theorem, there are roughly $\frac{N}{\log N}$ primes in $[1, N]$.

So there are about $\frac{2^k}{k}$ primes in $[1, 2^k]$, and $\frac{2^{k+1}}{k+1}$ in $[1, 2^{k+1}]$

So in our interval $[2^k, 2^{k+1}]$ there are

$$\frac{2^{k+1}}{k+1} - \frac{2^k}{k} =$$

By the prime number theorem, there are roughly $\frac{N}{\log N}$ primes in $[1, N]$.

So there are about $\frac{2^k}{k}$ primes in $[1, 2^k]$, and $\frac{2^{k+1}}{k+1}$ in $[1, 2^{k+1}]$

So in our interval $[2^k, 2^{k+1}]$ there are

$$\frac{2^{k+1}}{k+1} - \frac{2^k}{k} = \frac{2^k}{k}$$

primes.

By the prime number theorem, there are roughly $\frac{N}{\log N}$ primes in $[1, N]$.

So there are about $\frac{2^k}{k}$ primes in $[1, 2^k]$, and $\frac{2^{k+1}}{k+1}$ in $[1, 2^{k+1}]$

So in our interval $[2^k, 2^{k+1}]$ there are

$$\frac{2^{k+1}}{k+1} - \frac{2^k}{k} = \frac{2^k}{k}$$

primes.

So if we pick a random number in $[2^k, 2^{k+1}]$

By the prime number theorem, there are roughly $\frac{N}{\log N}$ primes in $[1, N]$.

So there are about $\frac{2^k}{k}$ primes in $[1, 2^k]$, and $\frac{2^{k+1}}{k+1}$ in $[1, 2^{k+1}]$

So in our interval $[2^k, 2^{k+1}]$ there are

$$\frac{2^{k+1}}{k+1} - \frac{2^k}{k} = \frac{2^k}{k}$$

primes.

So if we pick a random number in $[2^k, 2^{k+1}]$, the probability that it is a prime is $\frac{2^k/k}{2^k} = \frac{1}{k}$.

Probability that a random number in $[2^k, 2^{k+1}]$ is not prime is $1 - \frac{1}{k}$.

Probability that a random number in $[2^k, 2^{k+1}]$ is not prime is $1 - \frac{1}{k}$.

So pick k random numbers. Then the probability that none of them are primes is

$$\left(1 - \frac{1}{k}\right)^k \approx \frac{1}{e} = 0.37$$

i.e. the probability that at least one of them is a prime is 0.63.

Probability that a random number in $[2^k, 2^{k+1}]$ is not prime is $1 - \frac{1}{k}$.

So pick k random numbers. Then the probability that none of them are primes is

$$\left(1 - \frac{1}{k}\right)^k \approx \frac{1}{e} = 0.37$$

i.e. the probability that at least one of them is a prime is 0.63.

What if we instead picked k^2 random numbers?

Probability that a random number in $[2^k, 2^{k+1}]$ is not prime is $1 - \frac{1}{k}$.

So pick k random numbers. Then the probability that none of them are primes is

$$\left(1 - \frac{1}{k}\right)^k \approx \frac{1}{e} = 0.37$$

i.e. the probability that at least one of them is a prime is 0.63.

What if we instead picked k^2 random numbers?

We can make our probabilities arbitrarily close to 1, and still only sample a teeeeeny fraction of the numbers in $[2^k, 2^{k+1}]$ for large k.

So what was the point of all this?

So what was the point of all this?

Well, P = BPP tells us that we should be able to derandomize this algorithm.

So what was the point of all this?

Well, P = BPP tells us that we should be able to derandomize this algorithm.

This means that there exists a really fast deterministic way to generate primes in any interval in polynomial time!!!!

The End. Beep Boop.