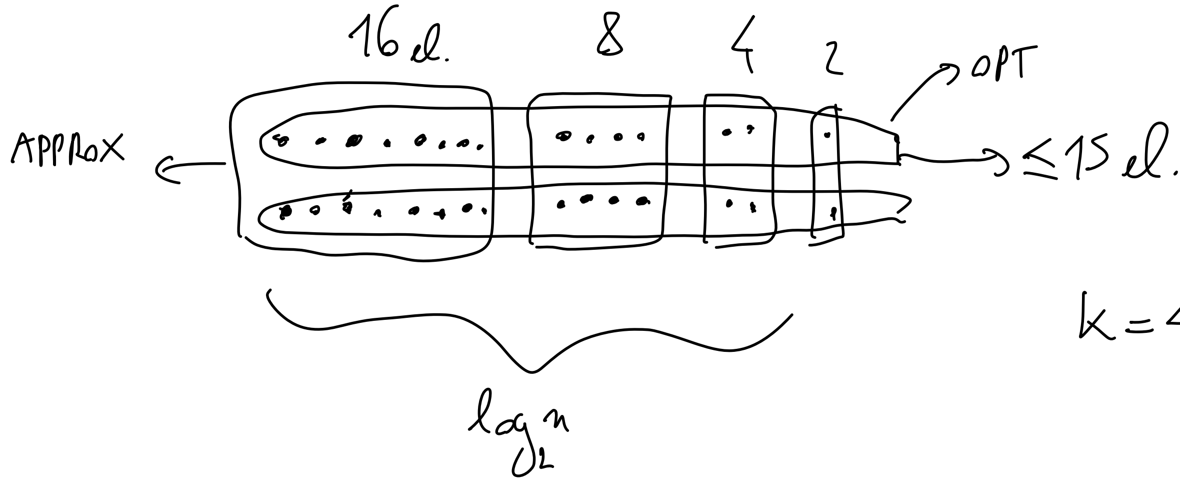
Exercise: show that there is an input $I = (X, F)$ on which APPROX_SET_COVER achieves an approximation ratio of $\Theta(\log n)$



$k = 4$

X has $n = 2^{(k+1)} - 2$ elements for some $k \in \mathbb{N}$

F has
1) k pairwise disjoint sets $S_1, \dots, S_k$ with sizes $2, 4, \dots, 2^k$
2) two additional disjoint sets T_0, T_1 each of which contains half of the elements from each S_i

APPROX_SET_COVER $\longrightarrow S_k, S_{k-1}, \dots, S_1$

OPT $\longrightarrow T_0, T_1$

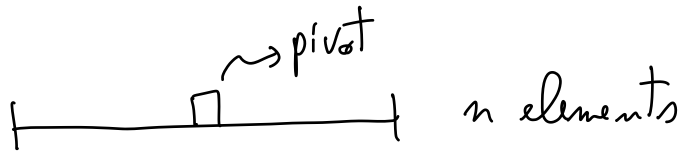
ratio: $\frac{k}{2} = \Theta(\log n)$

Randomized Algorithms

are algorithms that may do random choices ... → e.g. flip a coin

... but why?

Example 1 : Randomized Quicksort



$$T_{QS}(n) = O(n^2)$$

RQS : choose pivot at random. This hides the worst-case inputs from the adversary

$$E\left[T_{RQS}(n)\right] = O(n \log n)$$

Example 2 : verify polinomial identities

check whether

$$\underbrace{(x+1)(x-2)(x+3)(x-4)(x+5)(x-6)}_{H(x)} \stackrel{?}{\equiv} \underbrace{x^6 - 7x^3 + 25}_{G(x)}$$

obvious algorithm : transform $H(x)$ in canonical form $\sum_{i=0}^{d=6} c_i x^i$ and then verify whether all the coefficients c_i of all monomials are equal

d = maximum degree

complexity : $O(d^2)$

a faster algorithm :

- choose a random integer r
- compute $H(r)$ // $O(d)$
- compute $G(r)$ // $O(d)$
- if $H(r) = G(r)$ then return YES
- else return NO

Does it work ?

example : $r = 2$

$H(2) = 0$

$G(2) = 33$ $\Rightarrow H(x) \neq G(x)$

what if $H(r) = G(r)$?

example : $x^2 + 7x + 1 \equiv (x+2)^2$

$r=2$: $4+14+1=19$ $\neq$
$4^2=16$

$r=1$: $1+7+1=9$ $=$
$3^2=9$

→ alg. returns YES, but this is <u>wrong</u>

unlucky choice of r

If the equation is correct, the algorithm is always correct
Otherwise, the algorithm returns the wrong answer only if r is a root of the polynomial $F(x) = G(x) - H(x) = 0$

If $r \in \{1, 2, \ldots, 100d\}$ when d is the max degree in $F(x)$, then

$$\Pr(\text{algorithm fails}) \leq \frac{d}{100d} = \frac{1}{100}$$

small, but still not satisfactory

How to reduce the probability of error?

- run the algorithm 10 times
- if YES in all the 10 runs then return YES
- else return NO

Now

$$\Pr(\text{algorithm fails}) \leq \left(\frac{1}{100}\right)^{10} = 10^{-20} < 2^{-64}$$

it's easier that your computer returns a wrong answer because it gets hit by a cosmic radiation that makes some bits flip!