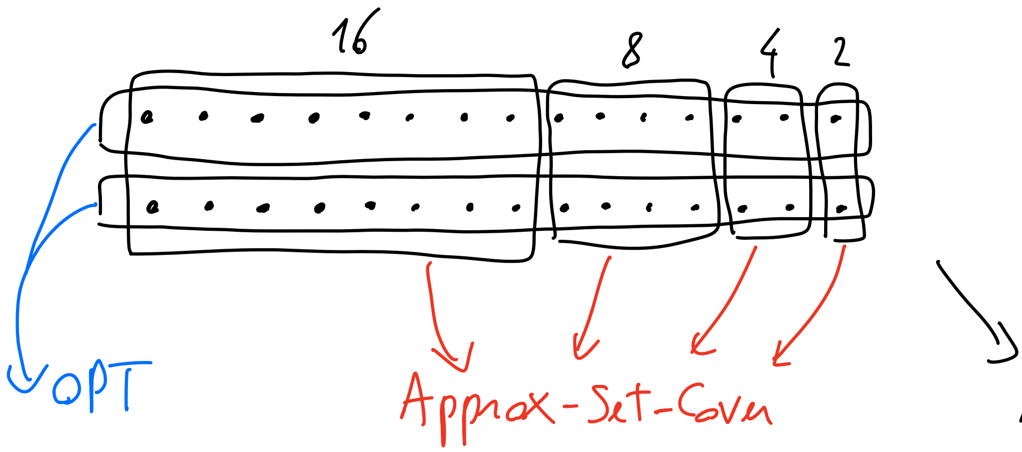
Exercise: show that there is an input $I = (X, F)$ on which Approx-Set-Cover achieves an approximation ratio of $\Theta(\log n)$



$n = 30$
$k = 4$

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 $2, 4, \dots, 2^k$

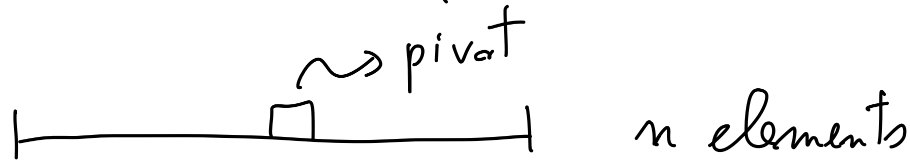
2) two additional disjoint sets T_0, T_1 each of which contains half of the elements from each S_i

Approx-set-cover $\longrightarrow S_k, S_{k-1}, \dots, S_1$
OPT $\longrightarrow T_0, T_1$
ratio: $k/2 = \Theta(\log n)$

Randomized Algorithms

are algorithms that may do <u>random</u> choices ....
.... but why? Seems paradoxical!

<u>Example 1</u> : Randomized Quicksort



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

RQS: choose the pivot <u>at random</u>

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

more discussion
in Further reading

This hides the worst-case inputs
from the adversary

↳ does no longer know
algorithm's moves
in advance

Example 2 : verifying polinomial identities

check whether

$$(x+1)(x-2)(x+3)(x-4)(x+5)(x-6) \overset{?}{\equiv} x^6 - 7x^3 + 25$$

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

obvious algorithm : transform $H(x)$ in canonical form $\sum_{i=0}^{d=6} c_i x^i$ and 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 // new operation
- 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: $x^2 + 7x + 1 \stackrel{?}{\equiv} (x+2)^2$

$r = 2$: $19 \neq 16$

$r = 1$: $9 = 9$ → alg. returns YES, but it 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 $r \in \{1, 2, \dots, 100d\}$ where 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 times then return YES
- else return NO

Now

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

2^{-64} is comparable to the probability of a hardware error in your computer caused by cosmic radiation (quoting D. Knuth). So, this alg. is correct for all practical purposes.

Classification of randomized algorithms

1) rand. alg. that never fail → "LAS VEGAS" alg.
e.g. randomized Quicksort

$$\forall i \in I, \; A_R(i) = \overset{\text{solution}}{s} \text{ s.t. } (i, s) \in \Pi$$

obs.: s may not be the same $\forall i$

randomness come into play in the analysis of the complexity

$\forall n$, $T(n)$ is a random variable, of which we usually study $E[T(n)]$ or

$$\Pr\left(T(n) > c \cdot f(n)\right)$$

$\hookrightarrow \leq \frac{1}{n^k}$ $\rightarrow T(n) = O(f(n))$ "with high probability"

space of probabilities = random choices made by the algorithm

(Do not confuse this with the probabilistic analysis of deterministic algorithm, where the space of probabilities = distribution of the inputs)

2) rand. alg. that <u>may fail</u> $\rightarrow$ "MONTE CARLO" alg.

e.g. verifying polimomial identities

$i \in I$ it's possible that $A_R(i) = s$ s.t.

$(i, s) \notin \Pi$

We study $Pr\left((i,s) \notin \Pi\right)$ as a function of

$n = |i|$ $\longrightarrow$ family of random variables

moreover, even $T(n)$ may be a random variable for decision problems, these alg. can be divided into

- <u>one-sided</u> : may fail only on one answer
- <u>two-sided</u> : may fail in both answers

We'll see 1 LAS VEGAS and 1 MONTE CARLO

$\downarrow$ Randomized Quicksort

$\downarrow$ Karger's alg. for minimum cut