Exercise 1: Fix $k \in \mathbb{N}$

2-SUM: find i, j st $x_i + x_j = S$.

$\boxed{\text{k-SUM problem}}$

input: $x_1, \dots, x_n \in \mathbb{N}$ and target $S \in \mathbb{N}$

output: is it possible to find a subset $I \subseteq \{1, \dots, n\}$ of size k such that $\sum_{i \in I} x_i = S$?

Note: k is fixed but n (# of x_i) is not (part of the input)

Q1: polynomial time algorithm for k-SUM (for any fixed k)

try every set $I \subseteq \{1, \dots, n\}$ of size k and then check whether $\sum_{i \in I} x_i = S$.

complexity: • $\binom{n}{k} \le n^k$ listing I

• check: $O(k)$ to sum

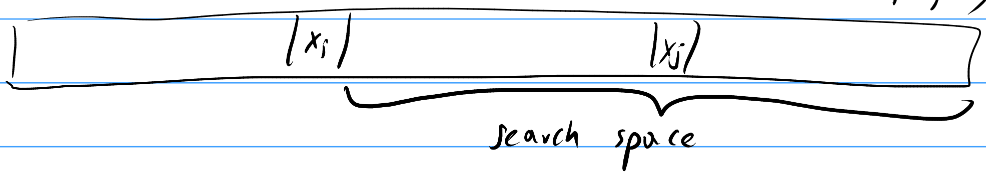
$\Big\}$ $O(n^k \cdot k) = O(n^k)$ polynomial in n.

Q2: 2-SUM we have a $O(n^2)$ algorithm

1) sort the x_i (increasing)

2) go through the x_i, binary search for $S - x_i$ in the sublist $x_{j+1}, \dots, x_n$

$O(n)$

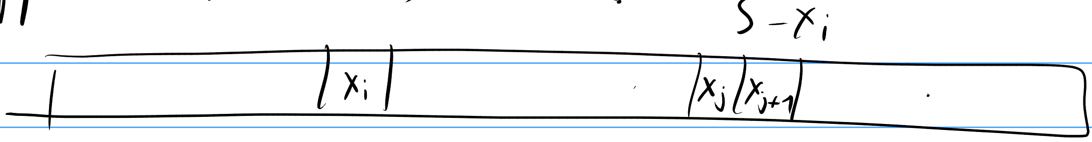


Complexity: 1) $O(n \log n)$

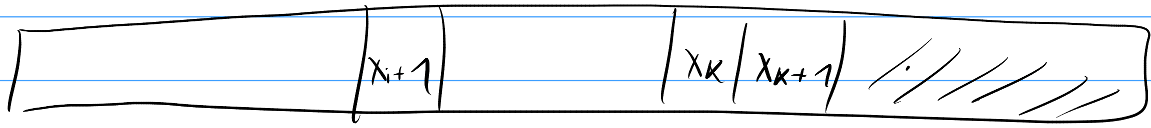
2) repeat n times, a binary search $O(\log n)$
$= O(n \log n)$.

$= O(n \log n)$.

suppose we have this situation:

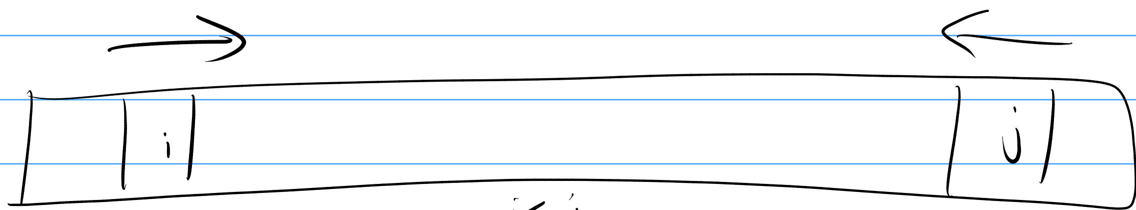


$x_i + x_j \leq S$ but $x_i + x_j > S$



$x_i \leq x_{i+1}$ want $x_{i+1} + x_k \leq S$ but $x_{i+1} + x_k > S$

$k \leq j$.



$i < j$

algo: $\boxed{x_i + x_j = S}$

$j \leftarrow n$, $i \leftarrow 1$
while $i < j$
  while $i < j$ and $x_i + x_j > S$
    $j \leftarrow j - 1$))
  check if $x_i + x_j = S$
  if yes, STOP
  else $i \leftarrow i + 1$))

complexity:

$O(n)$ because
- i can only increase n times
- j —— decrease n times

Q3: 3-SUM we have a $O(n^3)$ solution

- $O(n^2 \log n)$: 1) sort x_i $O(n \log n)$
  2) for all $i \neq j$ $O(n^2)$
  binary search $S - x_i - x_j$ $O(\log n)$

$= O(n \log n + n^2 \log n) = O(n^2 \log n)$

- $O(n^2)$: 1) sort x_i
  2) For all K, we are looking i and j $O(n)$
  st $x_i + x_j + x_K = S \iff x_i + x_j = \underline{S - x_K}$
  apply the above algo to find i, j
  st $x_i + x_j = S - x_K$ $O(n)$

$$O(n \log n + n^2) = O(n^2)$$

__Exercise 2:__
- $Mul(n) =$ complexity multiply degree n polynomials
- $Mul(n_1) + Mul(n_2) \leq Mul(n_1 + n_2)$ (1)
- $Mul(mn) \leq m^2 Mul(n)$ (2)

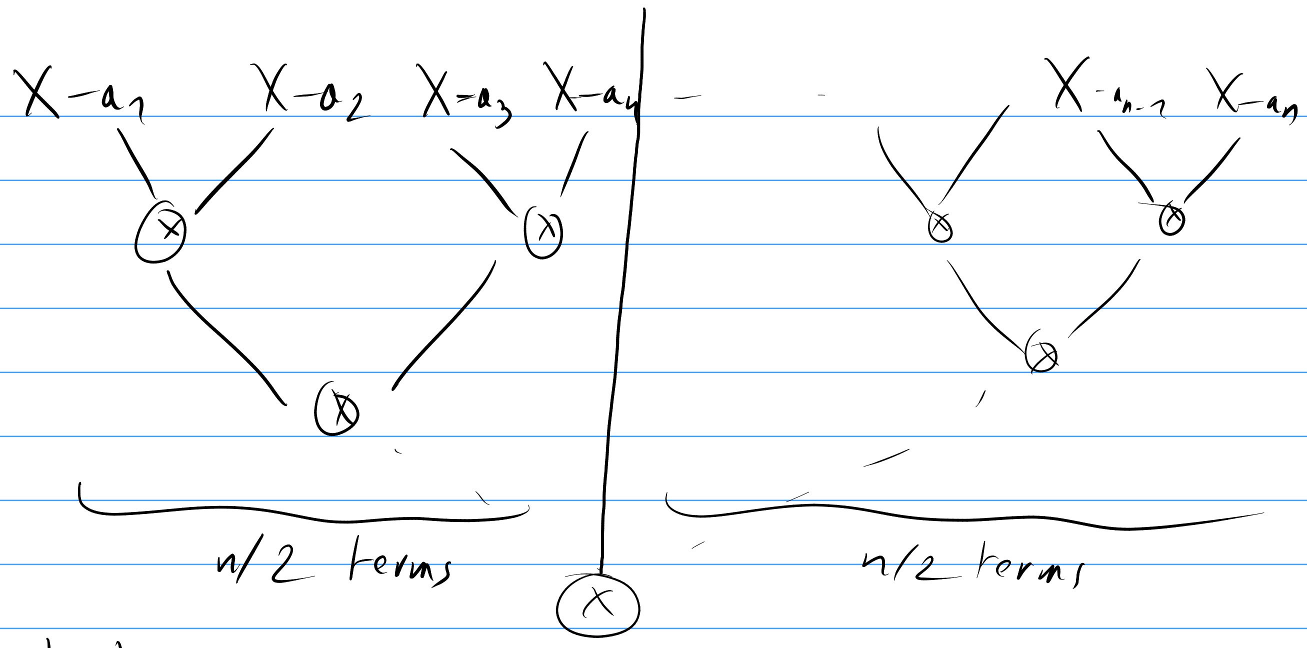
For simplicity, $n =$ power of 2.

__Q1__: given $a_1, \dots, a_n$, compute $\underbrace{(X - a_1) \cdots}_{K \text{ term } = \deg K} (X - a_n)$ in $O(Mul(n) \log n)$ op.

__naive__: start with $P_0 = 1$, define $P_{K+1} = P_K \cdot \underbrace{(X - a_K)}_{\deg 1}$
$\deg(P_K) = K$
$P_K \to P_{K+1}$: complexity $O(Mul(K))$

__total__: $O\left(\sum_{i=1}^{n} Mul(i)\right) \longrightarrow O(n \cdot Mul(n))$ bad
$\xrightarrow{(1)} O(Mul(\Sigma i)) = O(Mul(n^2))$ bad

__better:__



divide-and-conquer

Algo$(a_1, \dots, a_n)$

- if $n=1$ return $X-a_1$
- $P \leftarrow \text{Algo}(a_1, \dots, a_{\lfloor n/2 \rfloor})$
- $Q \leftarrow \text{Algo}(a_{\lfloor n/2 \rfloor + 1}, \dots, a_n)$
- return $P \times Q$ (using Mul)

Here $n =$ power of 2

Complexity: define $T(n)$ the complexity of Algo$(a_1, \dots, a_n)$

$$T(n) \leq 2T(n/2) + \text{Mul}(n/2) \qquad T(1) = 0$$

$$T(n) \leq 2\left(2T(n/4) + \text{Mul}(n/4)\right) + \text{Mul}(n/2)$$

$$\leq 4\,T\left(\tfrac{n}{4}\right) + 2\,\text{Mul}\left(\tfrac{n}{4}\right) + \text{Mul}\left(\tfrac{n}{2}\right)$$

$$\vdots$$

$$\leq n \cdot T(1) + \underbrace{\tfrac{n}{2}\text{Mul}(1) + \tfrac{n}{4}\text{Mul}(2) + \dots + 2\,\text{Mul}\left(\tfrac{n}{4}\right) + \text{Mul}\left(\tfrac{n}{2}\right)}_{\text{how many terms? } \log(n)}$$

we have terms like $2^k \text{Mul}\left(\frac{n}{2^k}\right)$ express in $\text{Mul}(n)$

$\leq \text{Mul}(n)$ why?

↳ by (1) $2\,Mul\left(\frac{n}{2}\right) \leq Mul(n)$ $\left[n_1 = n_2 = \frac{n}{2}\right]$

$2\,Mul\left(\frac{n}{4}\right) \leq Mul\left(\frac{n}{2}\right)$ $\left[\quad " \quad " \quad \frac{n}{4}\right]$

↳ $4\,Mul\left(\frac{n}{4}\right) \leq Mul(n)$

etc

in total: $T(n) \leq O\left(\log(n) \cdot Mul(n)\right)$

Master theorem: $T(n) \leq \underset{\underset{a}{\shortparallel}}{2} \cdot T\left(\underset{\underset{b}{\shortparallel}}{\frac{n}{2}}\right) + \underset{\underset{f(n)}{\shortparallel}}{Mul(n)}$

$c = \log_b(a) = \log_2(2) = 1$

3 cases: • $f(n) = O(n^c) = O(n)$? X

• $f(n) = O(n^c \log^k n) = O(n \log^k n)$?

by course $\boxed{Mul(n) = O(n \log n)}$ $k = 1$

Master: $T(n) = O(n^c \log^{k+1} n) = \boxed{O(n \log^2 n)}$

we cannot conclude that $T(n) = O(Mul(n) \log(n))$

because it would require $Mul(n) = \Omega(n \log(n))$

We could conclude with a generalized version of the Master theorem.

notations: $C = T$, $m = p = 2$, $f = Mul$

$\exists\, q, r$ st $q \leq \frac{f(pn)}{f(n)} \leq r$

$\exists\, q, r$ st $q \leq \frac{Mul(2n)}{Mul(n)} \leq r$

here: we can take $q = 2$, $r = 4$

$2\,Mul(n) \leq Mul(2n)$

by (1) $[n_1 = n_2 = n]$

$Mul(2n) \leq 4\,Mul(n)$

by (2) $[m = 2]$

We can apply the "even more general" theorem, we are in the case where $q = 2 = m$

so $T(n) = O(f(n)\log n) = O(\mathrm{Mul}(n)\log n)$

Q2: P of degree n

compute $P(a_1), \dots, P(a_{n/2})$
from $R(a_1), \dots, R(a_{n/2})$

where R = remainder of P divided by $(X-a_1)\cdots(X-a_{n/2})$

means: $P = (X-a_1)\cdots(X-a_{n/2})\,Q + R$ by def

$$P(a_i) = \underbrace{(a_i-a_1)\cdots(a_i-a_{n/2})}_{=0}Q(a_i) + R(a_i)$$
$$= R(a_i)$$

Q3: $P, a_1, \dots, a_n$, $\boxed{\deg(P) \simeq n}$

by Q2: for $i \leq \frac{n}{2}$, $P(a_i) = R(a_i)$, R = rem $P / (X-a_1)\cdots(X-a_{n/2})$

for $i > \frac{n}{2}$, $P(a_i) = R'(a_i)$ R' = rem $P / (X-a_{n/2+1})\cdots(X-a_n)$

two problems: eval R (R') at $n/2$ numbers
$\deg(R) = \deg(R') \leq n/2$

$\text{Eval}(P, a_1, \dots, a_n)$: $\longrightarrow$ returns $[P(a_1), \dots, P(a_n)]$

- if $n=1$: return $[P(a_1)]$
- $R \leftarrow \text{rem } P / \boxed{(X-a_1)\cdots(X-a_{n/2})}$
- $R' \leftarrow \text{rem } P / \boxed{(X-a_{n/2})\cdots(X-a_n)}$
- return $\underbrace{\text{Eval}(R, a_1, \dots, a_{n/2})}_{[R(a_1), \dots, R(a_{n/2})]} \underset{\text{concat}}{+} \underbrace{\text{Eval}(R', a_{n/2+1}, \dots, a_n)}_{[R'(a_{n/2+1}), \dots, R'(a_n)]}$

$\deg P = n$

$\deg (X-a_1)\cdots(X-a_{n/2}) = n/2$

$[R(a_1), \dots, R(a_{n/2})] + [R'(a_{n/2+1}), \dots, R'(a_n)]$

$(Q2) = [P(a_1), \dots, P(a_{n/2})] + [P(a_{n/2+1}), \dots, P(a_n)]$

complexity: remark: without loss of generality, we always have $\deg(P) \leq n$

why? true after one iteration because of division. for the first call, true by assumption ($\deg(P) = n$)

$T(n)$ = complexity of $\text{Eval}(P, a_1, \dots, a_n)$ with $\deg(P) \leq n$

$$T(n) \leq \underbrace{2\,\text{Div}(n)}_{\text{cost division}} + \underbrace{2\,\text{Mul}(n/2)\log(n/2)}_{\text{cost of comp } (X-a_1)(\dots)(X-a_{n/2}) \text{ by Q1}} + 2T\left(\frac{n}{2}\right)$$

- $\log(n/2) \leq \log(n) - 1$
- $2\,\text{Mul}\left(\frac{n}{2}\right) \leq \text{Mul}(n) \quad (1)$
- $2\,\text{Div}(n) \leq 2\,\text{Mul}(n)$

$\text{Div}(n) = O(\text{Mul}(n))$

$$T(n) \leq 2T\left(\frac{n}{2}\right) + \text{Mul}(n)\log(n)$$

therefore $T(n) \leq O(\text{Mul}(n)\log^2 n)$

We can do better: by avoiding to recompute $(X-a_1)\cdots(X-a_{n/2})$

$$T(n) \leq 2\,\text{Div}(n) + \text{Mul}(n) + 2T\left(\frac{n}{2}\right)$$

$$T(n) = O(\text{Mul}(n)\log(n)).$$