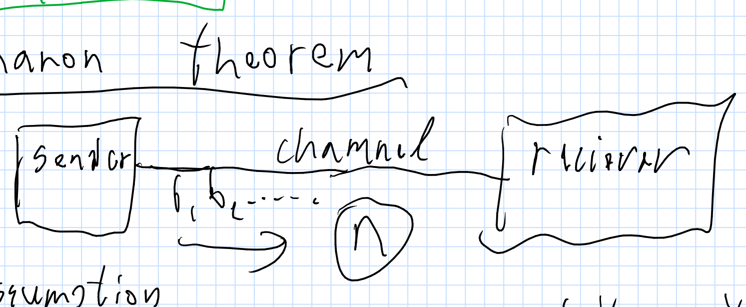
12: Union find and Shanon

3/11/2021

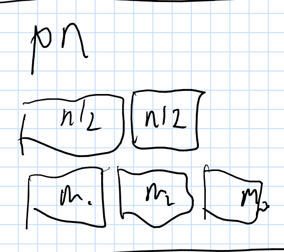
Shanon theorem



Assumption

The channel flips a bit with probability p. p is a parameter.

pn

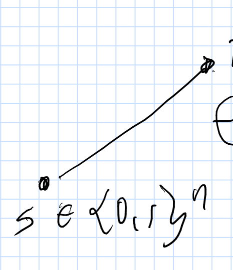


$\geq n(1-H(p)-\delta)$ Shanon theorem

$p = \frac{1}{2} \simeq$ impossible

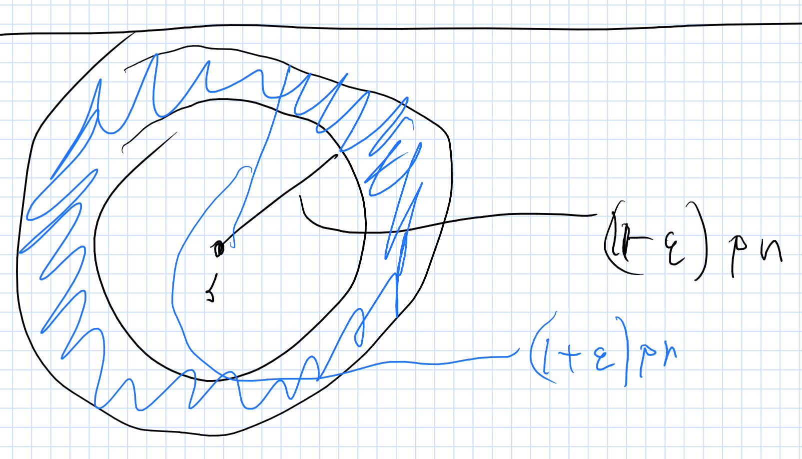
$p = 1$

$H(p)$



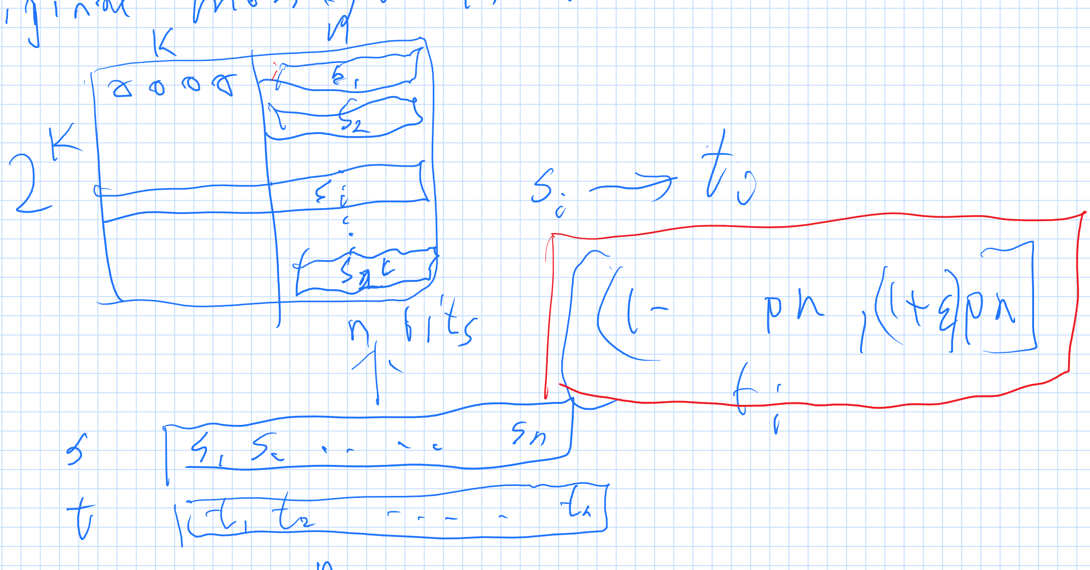
$E[d_H(s,t)] = pn$

Lemma $P[(1-\varepsilon)pn \leq d_H(s,t) \leq (1+\varepsilon)pn] \geq 1 - \frac{1}{n^{O(1)}}$

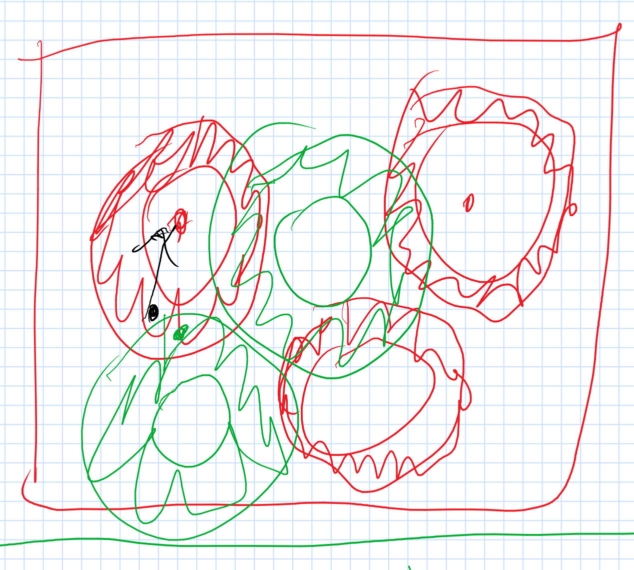


$K = (1-h(p))n$

original message is k bits



$d_H(s,t) = \sum_{i=1}^{n} |s_i \neq t_i|$



$K = (1-H(p))n$

Union-Find

$\log^* n$

How many times you have to apply log before you get a number smaller than 1

log in base 2.

$\log 2 = 1$

$\log^* 2 = 1$

$\log^* 4 = 2$ $\log\log 4 = \log 2 = 1$

$\log^* 16 = \log^* 2^4 = 1 + \log^* 4 = 3$

$\log^* 2^{2^2}$

$\log^* 2^{16} = \log^* 65536 = \log^* 2^{2^{2^2}} = 4$

$\log^* 2^{65536} = 5$

10^{6500} ←

$Tower(i) = 2^{2^{2^{\dots^2}}}$ i times

$\log^*(Tower(i)) = i$

$f_1(x) = 2x$

$f_2(x) = \underbrace{f_1(f_1(f_1 \dots (f_1(1))))}_{k \text{ time}} = 2^x$

$f_3(x) = \underbrace{f_2(f_2(f_2(\dots f_2(1))))}_{k \text{ times}} = tower(x)$

$f_4(x) = \underbrace{f_3(f_3(f_3 \dots f_3(1)))}_{x \text{ times}}$

$f_i(x) = \underbrace{f_{i-1}(f_{i-1}(\dots f_{i-1}(1)) \dots)}_{x \text{ times}}$

$g(i) = f_i(i)$ ← Ackermann's function

$g(4) = tower(65536)$

$t = g^{-1}(\ell)$ minimum t s.t. $g(t) \geq \ell$

$g^{-1}(n) << \log^* n$ ← n

Theorem

Consider a union-find data-structure implemented using reverse trees with

- union by rank
- path compression.

Then a sequence of m operations over n elements take

$O((m+n)\log^* n)$ Tarjan

$O((m+n)5)$ — five

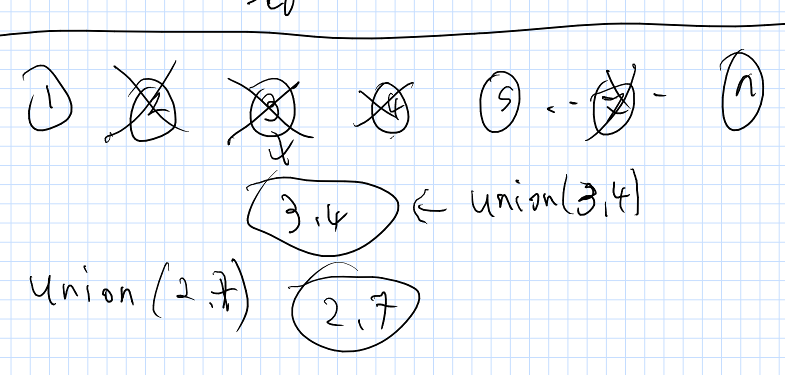
$\Theta((m+n)g^{-1}(n))$ ←

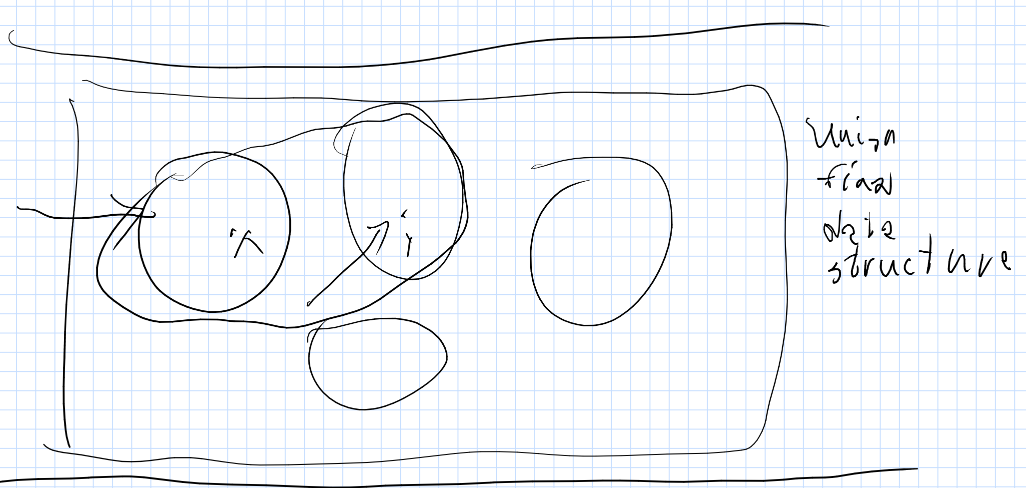
$1, 2, \dots, n$

$create\_set(x) = \{x\}$

$find(x) \equiv$ returns the set that contains x

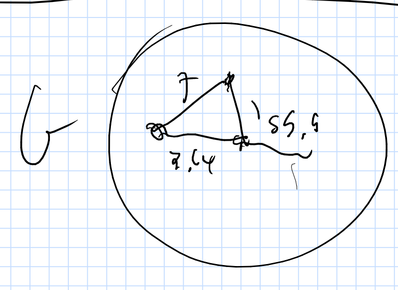
$Union(x,y) \equiv$ merge the two sets that contains x and y and merge them into a single set





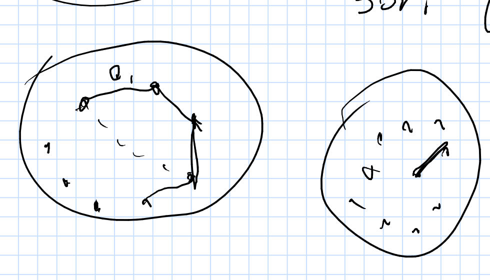
Union find data structure

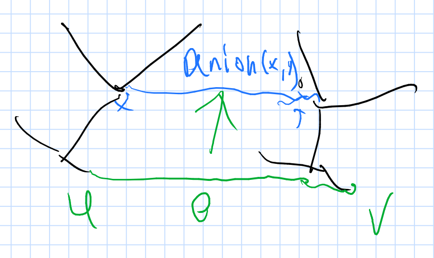
Computing MST by Kruskal algorithm



$e_1, e_2, \dots e_m$

sort $O(m\log m)$



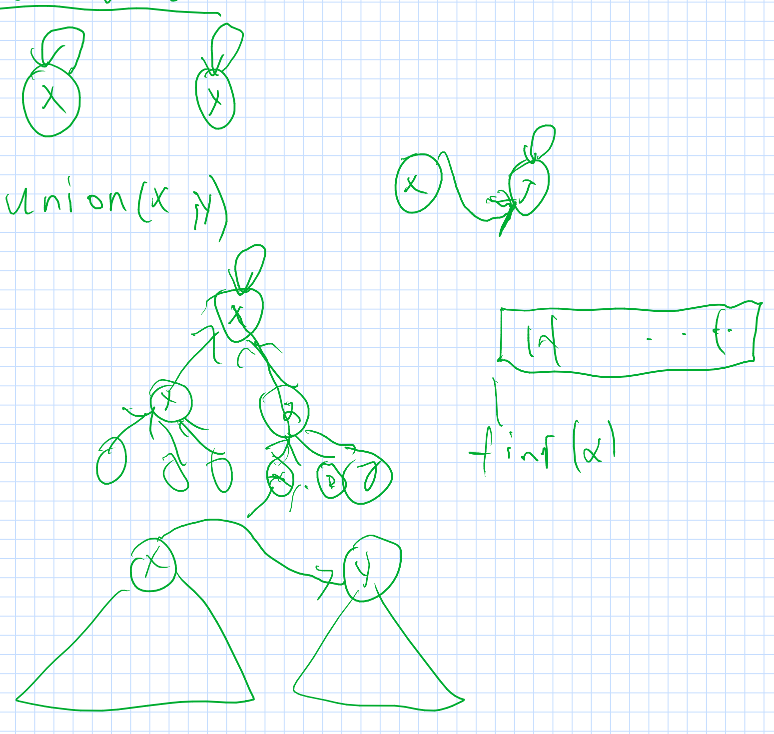


n element

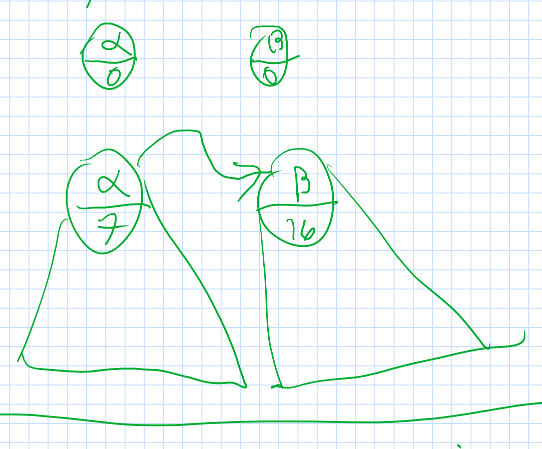
n union operation

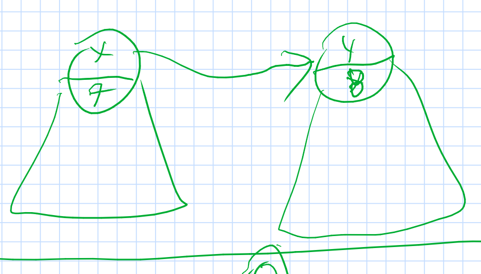
m find operations

Reverse trees

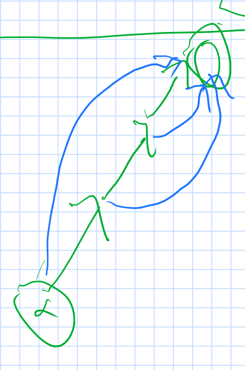


- Union by rank





union



path compression

find(α)

Union by rank