

Notes

9/29/26

• Indivisible Goods (NO CRUMBS!!!)

EF1 - Envy Free up to 1 item

- can envy someone, but if you delete their max value item, no longer envy them
- ie. for every player, the amount they envy another player is at most the maximum item in their bundle $\rightarrow$ ALWAYS GUARANTEEABLE

EFX

- if I delete any item I no longer envy
- always exists for 2 players, 3 players as well if players have w/ additive valuations
- ie the amount to which any player i envies another player j is at most the minimum valued item of player j .

$$EFX \gg EF1$$

EF1 is pretty weak

Maximin Share:

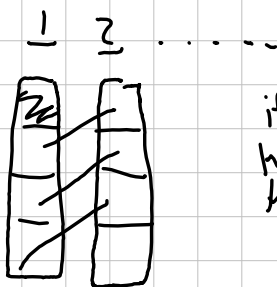
- if each player were to divide the bundles 'evenly', what would be the worst bundle
- we partition the set into n bundles, then what is the Max Min bundle aka. when would the worst bundle valuation be maximized?
- changes for each player, since each player values differently

• Max Nash Welfare

$$\max \prod_i V_i(A_i)$$

EF1:

Each player takes a turn picking most valued item until no items left



if we remove P_1 's first choice, then P_2 values his rd 1 choice over rd 2 choice of P_1 , since if he valued it more, he would have taken it

- P_1 in that sense, becomes the "n+1"th player

- EF1 holds

EFX: to prove EFX, $P_1 - P_2 < \min(P_2)$

$$P_2 - P_1 < \min(P_1)$$

- can also remove the min, and give it to the envious player

NP-Hardness

$P = \{ \text{the class of prob. that can be solved in polynomial amt of time} \}$
 a problem is just a set of strings that can be 'solved' in P time
 if membership can be determined

ex: Sorted = $\{ \langle x_1, \dots, x_n \rangle \mid x_1 \leq x_2 \leq \dots \leq x_n \}$

Sum = $\{ \langle x_1, \dots, x_n, T \rangle \mid \sum_{i=1}^n x_i = T \}$

Partition = $\{ \langle x_1, \dots, x_n \rangle \mid \exists S \text{ s.t. } \sum_{x \in S} x = \sum_{x \notin S} x \}$

$NP = \{ \text{the class of problems that I can verify a solution to in polynomial time} \}$

$$P \subseteq NP$$

- open question is $P = NP$

NP-Hardness shows some problems are obj. harder than others,
 rather than quantifying 'hard'

Claim: if I have a polynomial-time alg for subset-sum, then I can
 make a alg. for partition

Given an input for partition, $(x_1, \dots, x_n)$

let $S = \sum_{i=1}^n x_i$, let $T = \frac{S}{2}$

- just use subset-sum Alg on $(x_1, \dots, x_n, T)$

- this is a Reduction, which is a mapping that takes inputs for
 one problem (A) and computes/equivalent inputs for another prob B,
 in polynomial time

- called a Polynomial-Time Reduction

$$A \leq_p B$$

Notes 4/24

- Now can quantify hardness, because if we get an alg for B, it instantly works for A.
- We call some alg C NP-hard, if $\forall \text{DENP}, D \leq_p C$
 - if you get an alg for C, it works for every D.
- We say B is NP-complete if in NP, B is NP-hard