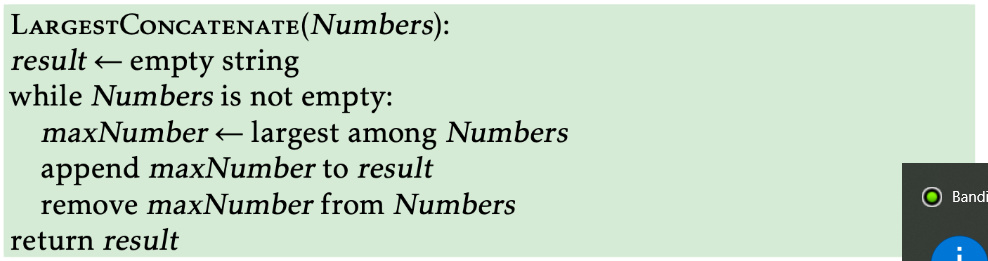
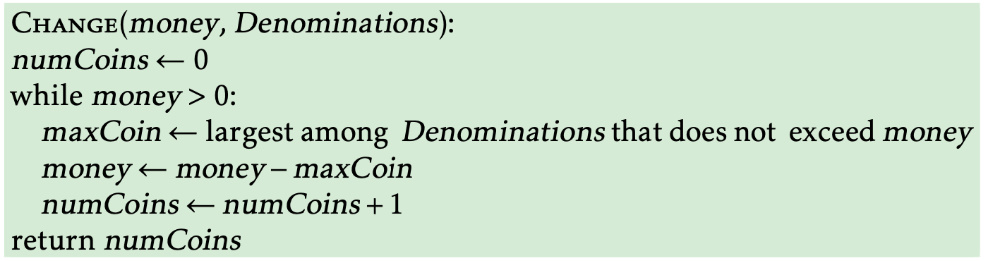
**Greedy Alghoritm**

* Algoritmul Greedy ofera o solutie bucata cu bucata la fiecare pas si alege cea mai profitabila bucata din cele avute la momentul dat.
* El se bazeaza pe faptul ca se alege o decizie din informatia avuta la un moment dat, fara a se gandi care ar putea fi efectele decizie luate in viitor sau daca rezultatul final va fi optim. Pe el il intereseaza ca rezultatul sa fie optim la momentul alegerii.
* Odata ce s-a ales o solutie optima dintr-o multime, ea e eliminata din multime si apoi din multimea ramasa se cauta alta solutie optima.
* De ex, fie ca avem un String cu numerele 2 3 9 2 2 si vrem sa cream cel mai mare numar cu ele. Algoritmul va fi asa:



* Fiecare iteratie ofera cate o bucata din solutie, si anume bucata optima, adica cel mai mare numar ce se afla in Numbers in acel moment, apoi o elimina si cauta solutia optima din cele ramanse si tot asa.
* Exemplul2: sa zicem ca vrem sa aflam cate monede de 10 5 si 1 ne trebuie pentru a obtine suma de bani data asa ca sa folosim nr minim de monede:



* Totusi, daca ne conducem dupa aceste algoritmuri, vom da gres.
* De ex, daca avem numerele 2 21, logic cel mai mare nr cu ele creat, dupa algoritm de sus, va fi 212, desi corect e 221
* Si daca avem 8 monede ce trebuie obtine cu monede din 6 4 si 1, dupa algoritmul de sus am avea 3(6+1+1), dar corect e 2(4+4)

**Demonstrarea corectidunii algoritmului Greedy**

* Pentru fiecare problema de mai sus, exista cate un algoritm, insa mai putem crea cate unul pentru fiecare dintre ele, mai corect.
* Sa zicem ca tt avem 2 21.

Alegem amplasarea lor dupa cel mai mare si apoi urmatorul cel mai mare, adica 212.

Daca le schimbam cu locul, obtinem 221, care e mai mare ca 212

**Cand il folosim**

1. Daca o solutie optimala a problemei poate fi gasita alegand cea mai buna alegere la fiecare pas fara a lua in considerare pasii anteriori, atunci putem folosi acest algoritm**(Greedyng choice property)**
2. Daca solutia generala optima a problemei corespunde cu solutia optimala a subproblemelor**(Optimal substructure)**

Algoritmul

1. Multimea de solutii intai e goala
2. La fiecare iteratie, un item e pus in multimea de solutii pana se gaseste solutia finala
3. Daca itemul selectat e bun, e pastrat
4. Daca nu, este respins si niciodata reincercat