Linked list: oplati sa ked len pridavam prvky a nic s nimi dalej nerobim

Odpis – lebo len pridavam O(1) a nieje nutna restrukturalizacia , to by mohol byt aj explicit queue aby to bolo akoze ,,korektne“

Kamiony – lebo len pridavam tie iste dovody

Arraylist: oplati sa ked pristupujem k prvkom a prechadzam ich

Evidencia vozidiel – lebo ju potrebujem prechadzat a pristupovat k prvkom cez[i] co je O(1) a pri linked to je O(n). Prejst teda 5 prvkov pri linked znamena 5!. Vymazat stadial a to je O(n) pri oboch. Pridavat je sice O(1) pri linked a pri array O (n) ale len ked sa musi zvacsit inak to je asi O(1) takze v tomto pripade sa preto oplati radsej arraylist lebo ten ma pridavanie O(n) len ked sa potrebuje zvacsit inak je to O(1).

Dodavatelia – treba ich prechadzat a pytat sa na dane mnozstva to je O(1)vid metoda 12. Pridavat je tiez O(1) kym sa nemusi zvacsit ale na zaciatku mu nastavim nejaku dostatocnu velkost.

Sklad – lebo potrebujem ho prechadzat a vypisovat palety, ktore sa tam nachadzaju O(1), pri linked je daj prvok na indexe O(n). Potrebujem tam pridavat to je O(1) a len niekedy O(n) co sa ale aj tak stale lepsie oplati ako linkedlist.

Nezrealizovane palety – vypisanie, pristup k nim , vyhladavanie

Kamion ma arraylist paliet – lebo k nim pristupujem

Zasielka ma arraylist paliet – lebo k nim pristupujem

Prior.front:

mozuRozvazat – lebo tam potrebujem mat navrchu tie ktore maju najmensie opotrebovanie

Implicit Stack:

Vozidlo bude mat stack paliet, lebo sa stadial budu po rade vyberat a vlastne tym sa aj vymazu a to potrebujem. Implicit je robeny polom a vsetky operacie su O(1).