Véletlenített algoritmusok

Tegyük fel, hogy van két doboz (A,B), amely egyike 1000 Ft-ot tartalmaz, a másik üres. 500 Ft-ért választhatunk egy dobozt, amelynek a tartalmát megkapjuk.

A feladat megoldására két determinisztikus algoritmus használható vagy A-t választjuk vagy B-t. Mindkét algoritmus költsége a legrosszabb esetben 500 (azon input esetén, amelyben nem találtuk meg a pénzt).

Másrészt, ha egy olyan véletlenített algoritmust használunk, amely $1/2$ valószínűséggel választja mindkét ládát, akkor mindkét inputra az algoritmus költségének várható értéke $1/2 \cdot 500 + 1/2 \cdot -500 = 0$.

Ilyen esetben, ha az algoritmus használ véletlenül generált döntéseket, véletlenített algoritmusról beszélünk.

Ezen algoritmusok futása függ a véletlen döntésektől, így az algoritmus hatékonysága (a kapott eredmény vagy a futási idő) egy valószínűségi változó, amelynek a várható értékét szokás vizsgálni.

Véletlenített online algoritmusok esetén a versenyképesség definíciójában az $A(I)$ valószínűségi változó várható értékét használjuk. Az inputot generáláló ellenféltől függően 3 modellt definiáltak

- hanyag fellenfél: az inputot az algoritmus véletlen döntéseinek eloszlása ismeretében, de a döntések kimenetének ismerete nélkül kell megadja
- adaptív online ellenfél: az input generálása során mindig megkapja az online algoritmus döntéseinek kimenetét is, de az inputot az ellenfél is online oldja meg.
- adaptív offline ellenfél: az input generálása során mindig megkapja az online algoritmus döntéseinek kimenetét is, de az inputot az ellenfél offline az input végén oldja meg.

Síbérlés véletlenített algoritmus

Véletlenített algoritmus esetén az algoritmus költsége egy valószínűségi változó, a versenyképességi hányados definíciójában annak várható értékét hazsnáljuk.

Vegyük a következő véletlenített algoritmust a síbérlési feladatra (a sí vásárlási ára B egység). Az R algoritmus 1/2 valószínűséggel a 3/4B időpontig vár és utána vásárol, 1/2 valószínűséggel pedig a B időpontban.

Tétel Az algoritmus 15/8-versenyképes.

Bizonyítás: Vegyünk egy tetszőleges inputot, jelölje I. Azaz I napig síelünk. Különböztessük meg a következő eseteket.

- Ha I < 3/4B, akkor az optimális költség I, továbbá R költsége is I mindkét döntés esetén, így az $E(R(I))/OPT(I)$ arány 1.
- Ha $3/4B \leq I < B$, akkor az optimális költség I, továbbá R költsége vagy $B + 3/4B - 1$ vagy I. Tehát $E(R(I)) \leq 1/2 \cdot 7/4 \cdot B + 1/2 \cdot I$. Felhasználva, hogy $I \geq 3/4B$ kapjuk, hogy $E(R(I))/OPT(I) \leq (1/2 \cdot 7/4 \cdot B + 1/2 \cdot I)/I \leq 1/2 \cdot 7/4 \cdot 4/3 + 1/2 = 10/6$.
- Ha $B \leq I$, akkor $OPT(I) = B$. Az R algoritmus költsége pedig vagy $B + 3/4B - 1$ vagy $2B - 1$, így $E(R(I)) \leq 1/2 \cdot 7/4 \cdot B + 1/2 \cdot 2 \cdot B = 15/8B$. Következésképpen

  $E(R(I))/OPT(I) \leq 15/8$.

Lista karbantartás

BIT algoritmus Kezdetben minden elemhez hozzárendelünk egy bitet egymástól függetlenül egyenletes eloszlás alapján. Az i elemhez rendelt bit $b(i)$. Ezt követően, ha egy elemre meghívják a KERES algoritmust, akkor a hozzárendelt bitet megváltoztatjuk. Amennyiben a bit 0-ról 1-re változott, akkor az elemet a lista elejére visszük.

Tétel A BIT algoritmus 1.75 versenyképes.

TIMESTAP algoritmus Az x elemet a KERES(x) műveletet követően a lista legelső olyan eleme elé rakjuk a listában, amelyet legfeljebb egyszer kértek x utolsó kérése óta.

COMB algoritmus 4/5 valószínűséggel a BIT algoritmust használjuk, 1/5 valószínűséggel a TIMESTAP algoritmust.

Tétel A COMB algoritmus 1.6 versenyképes.

Lapozás

A véletlenített bélyegző (RB) algoritmust a következőképpen definiálhatjuk.

Az RB algoritmus

1. Ha a kért lap a memóriában van, akkor amennyiben még jelöletlen megjelöljük.
2. Ha a kért lap nincs a memóriában, és nincs már jelöletlen lap a memóriá- ban, akkor az összes jelölést töröljük. Ezt követően veszünk egy jelöletlen lapot egyenletes elszolás alapján a memóriából (az előző lépés miatt van ilyen) a kért lapot ennek a lapnak a helyére berakjuk, majd megjelöljük.

Tétel Az *RB* algoritmus $2H_k$-versenyképes hanyag ellenfél ellen.

Vegyünk egy tetszőleges I inputot, és rögzítsünk egy optimális offline algoritmus, amit jelöljön OFF. Az inputot bontsuk fázisokra ugyanúgy, mint a determinisztikus esetben. Ekkor ismét teljesül, hogy akkor kezdődik új fázis, amikor az *RB* algoritmus törli a bélyegeket a memóriában. Vegyük észre, hogy a fázisok nem függnek az algoritmus véletlen döntéseitől (a fázison belüli költsége igen).

Jelölje a fázisok számát n. Most vizsgáljuk meg, hogy egy adott i-re mennyi *RB* várható költsége az i-edik fázisban. Ennek meghatározásához a fázis során kért lapokat két halmazba osztjuk. Réginek nevezzük azokat a lapok, akik a fázis megkezdésekor *RB* memóriájában voltak, ezek számát jelölje r_i. A többi lapot (akik nem szerepeltek *RB* memóriájában) új lapnak nevezzük, ezek száma legyen u_i. Minden új lap hibát okoz, és egyetlen lap sem okoz egynél több hibát egy fázison belül, így azt kapjuk, hogy az új lapok által kapott hibák száma u_i.

Most vizsgáljuk meg, hogy mennyi a régi lapok által okozott várható hibák száma. Nyilvánvalón egy régi lap csak az első megjelenésénél okozhat hibát, pontosan, akkor ha a lapot az algoritmus már kirakta a memóriájából. Vizsgáljuk meg ennek a valószínűségét.

Tegyük fel, hogy egy régi lap első megjelenésénél, a memória tartalmaz x darab új lapot és y darab megjelölt régi lapot (ezeket már szerepeltek a fázisban). Ekkor a $k-y$ jelöletlen régi lap közül egyenletes eloszlás alapján x darab nem szerepel a memóriában, így annak a valószínűsége, hogy a kért lap nem szerepel, azaz hibát okoz $x/(k-y)$. Tehát a lap általi hibák számának várható értéke $x/(k-y) \leq u_i/(k-y)$.

Következésképpen a régi lapok által várható hibák száma legfeljebb $\sum_{j=0}^{r_i} u_i/(k-j)$. Így azt kapjuk, hogy az algoritmus várható költsége a fázis során legfeljebb

$$u_i + \sum_{j=0}^{r_i} u_i/(k-j) \leq u_i H_k.$$

.

Összefoglalva azt kaptuk, hogy $RB(I) \leq \sum_{i=1}^{n} u_i H_k$.

Most vizsgáljuk meg az optimális költséget. Ehhez legyen d_i azon lapok száma, amelyek az i-edik fázis előtt szerepelnek az offline memóriában, de nem szerepelnek *RB* memóriájában. Feltesszük, hogy $d_1 = 0$, azaz ugyanazzal a memóriával kezdenek az algoritmusok. Használjuk a d_{n+1} értéket is, ez az algoritmus befejezésekor fennálló érték.

Legyen az offline algoritmus költsége az i-edik fázisban OFF_i. Mivel az i-edik fázisban pontosan azokat a lapokat kérik, amelyek szerepelnek a végén *RB* memóriájában, ezért minden lap, ami a fázis végén ezektől különbözik egy hibát okoz az offline algoritmus számára. Következésképpen $OFF_i \geq d_{i+1}$.

Másrészt azon új lapok, amelyek nem voltak az offline algoritmus memóriájában a fázis előtt, a számára is hibát okoznak. Ezen lapok száma legalább $u_i - d_i$, így azt kapjuk, hogy $OFF_i \geq u_i - d_i$.

A két korlát alapján adódik, hogy $OFF_i \geq (u_i - d_i + d_{i+1})/2$. Összegezve ezeket az értékeket adódik, hogy

$$OFF(I) \geq (\sum_{i=1}^{n} u_i - d_1 + d_{n+1})/2 \geq ((\sum_{i=1}^{n} u_i)/2.$$

Játékelméleti háttér

Mátrixjátékok: A mátrixjátékok a kétszemélyes zérusösszegű játékok. Amennyit az A játékos nyer a B játékos elveszíti. A játékot megadja a C nyereségmátrix, a sorok A stratégiái, az oszlopok B stratégiái. Az N_{ij} mező A nyeresége, ha az i és j stratégiákat választják.

Mixelt stratégiák: A játékos nem egy stratégiát választ, hanem egy $p = (p_1, \ldots, p_n)$ valószínűségi eloszlást definiál a stratégiái halmazán. Hasonlóan B is egy $q = (q_1, \ldots, q_m)$ eloszlást választ. Ekkor $p_i \cdot q_j$ annak a valószínűsége, hogy a játékosok az i-edik és j-edik stratégiát választják. Ezért a nyereség várható értéke $p^T Cq = \sum_{i=1}^{n} \sum_{j=1}^{m} p_i C_{ij} q_j$

Kapcsolat az algoritmusokkal: Az A játékos választja a B online algoritmust, az A játékos generálja az I inputot. A nyerességmátrix az $A(I)/OPT(I)$, amit A maximalizálni, B minimalizálni akar. A B játékosnál a tiszta stratégia a determinisztikus algoritmusokat adja meg a mixelt a véletlenített algoritmusokat. A mixelt stratégiái a véletlenül generált inputoknak felelnek meg.

Minimax tételek a játékelméletben

Neumann Minimax Tétel Bármely C mátrix által megadott 2 személyes zérusösszegű játékra,

$$\max_p \min_q p^T Cq = \min_q \max_p p^T Cq.$$

Ha p fixált, akkor $p^T Cq$ egy lineáris függvénye q-nak, és minimalizált az által, hogy az a q_j 1 -re van állítva, amihez a legkisebb együttható tartozik ebben a lineáris függvényben. Tehát ha B ismeri A p eloszlását, akkor az ő optimális stratégiája tiszta lehet. Ez fordítva is igaz. Ez a minimax tétel egyszerűsített változatához vezet. Legyen e_k egy egységvektor, 1-essel a k-adik pozícióban, a többi érték legyen 0. Így a következő tételt kapjuk:

Loomis Tétel: Bármely 2 személyes zéróösszegű, C mátrix által adott játékra:

$$\max_p \min_j p^T Ce_j = \min_q \max_i e_i{}^T Cq.$$

Következmények online algoritmusokra (Yao elv)

Feltesszük, hogy a P online probléma olyan, hogy adott méretre véges számú inputtal, és véges számú determinisztikus algoritmussal rendelkezik. Ekkor a fentieknek megfelelően a versenyképessége egy mátrixjátékkal írható le.

Loomis tétele alapján azt kapjuk, hogy a legrosszabb input eloszlásra esetére véve a lehető legjobb determinisztikus algoritmust ugyanazt a versenyképességet tudjuk elérni, mint a legjobb lehetséges véletlenített algoritmussal a legrosszabb determinisztikus inputon. Ennek következménye a Yao elv, amit gyakran használnak alsó korlátok igazolására.

Yao elv Tehát azt kaptuk, hogy tetszőlegesen választott input eloszlásra nézve az optimális determinisztikus online algoritmus várható versenyképessége, alsó korlátot ad a véletlenített online algoritmusok versenyképességére.

Síbérlés alsó korlát

Adjunk a Yao elv alapján egy alsó korlátot a síbérlési problémát megoldó véletlenített algoritmusok versenyképességi hányadosára. Használjuk a következő valószínűségi input eloszlást. Az input legyen 1/2 valószínűséggel $T/2$ és 1/2 valószínűséggel $3/2 \cdot T$.

Az első input esetén az optimális költség T/2 a második esetében T. Yao elv használatá- hoz meg kell határoznunk azt az optimális determinisztikus algoritmusra az $A(I)/OPT(I)$ hányados várható értékét. Vegyünk egy tetszőleges determinisztikus online algoritmust. Ezt egyértelműen meghatározza egyetlen érték, ami azt adja meg hányadik napot követően vásárol sílécet, ha addig nem fejeződik be a síelés. Jelölje az algoritmust meghatározó értéket x.

Ha $x \leq T/2$, akkor az algoritmus költsége mindkét lehetséges inputon $x+T$. Tehát $E(A(I)/OPT(I)) = 1/2(x+T)/(T/2)+1/2(X+T)/T \geq 3/2$.

Ha $T/2 < x \leq 3/2T$, akkor az algoritmus költsége az első lehetséges inputra T/2 a másodikra $x+T$, így $E(A(I)/OPT(I))$ $1/2(T/2)/(T/2)+1/2(x+T)/T \geq 1/2+1/2\cdot 3/2 = 5/4$.

Ha $3/2T \leq x$, akkor az algoritmus költsége az első lehetséges inputra $T/2$ a másodikra $3/2T$, így $E(A(I)/OPT(I)) =$ $1/2(T/2)/(T/2)+1/2(3/2\cdot T)/T = 5/4$.

Következésképpen minden determinisztikus online algoritmusra teljesül, hogy az adott input eloszlás mellett $E(A(I)/OPT(I)) \geq 5/4$, így a Yao elv alapján adódik, hogy nincs véletlenített algoritmus, amelynek a versenyképességi hányadosa kisebb lenne, mint $5/4$.

Lapozás alsó korlát

Tétel Nincs olyan véletlenített algoritmus a lapozási problémára, amely versenyképességi hányadosa hanyag ellenfél ellen kisebb mint H_k.

Bizonyítás: A tétel bizonyításához a Yao elvet használjuk fel. Tehát vehetünk egy tetszőleges valószínűségi eloszlást, és az azáltal generált inputon kell megvizsgálnunk a legjobb determinisztikus online algoritmust.

Általában egy adott eloszlásra a legjobb determinisztikus algoritmus megtalálása igen nehéz feladat, így az ilyen esetekben gyakran használt ötlet olyan eloszlást választani, amelyen minden algoritmus ugyanazt a várható erdményt éri el.

Most is ezt tesszük. Vegyünk k+1 különböző lapot és legyen *I* kérések egy olyan sorozata, amelyben *n* darab kérés érkezik, melyek mindegyike egyenletes eloszlás alapján választ egy lapot a $k+1$ lap közül.

Definiáljuk a generált sorozat fázisait, az előző bizonyításokhoz hasonlóan. Az i-edik fázis az LFD algoritmus i-edik hibájánál kezdődik és a következő hibát megelőző lapig tart. Ekkor LFD költsége minden fázisban 1.

Most vizsgáljuk az adott fázis várható hosszát. Akkor következik be a következő hiba, amikor arra a lapra érkezik a kérés, amit LFD kitett a memóriából, de ezt megelőzően a többi lapra is kellett kéréseknek érkeznie.

Tehát egy fázis várható hossza 1-el rövidebb a legrövidebb olyan sorozat várható hosszánál, amelyben mind a k+1 lapra érkezik kérés. Másrészt pontosan az ilyen sorozatok várható hosszát vizsgálják a kupon gyűjtő problémában (a ládák, amibe a kupont rakjuk a lapok, amelyeket egyenletes eloszlás alapján generáltunk).

Következésképpen egy ilyen legrövidebb sorozat várható hossza $(k+1)H_{k+1}$. Azaz egy fázis várható hossza $(k+1)H_{k+1}-1 = (k+1)H_k$.

Most vegyük észre, hogy tetszőleges online algoritmust véve minden lépésben a hiba valószínűsége, így várható értéke $1/(k+1)$. Tehát az online algoritmus várható költsége egy fázisban $(k+1)H_k/(k+1) = H_k$.

Tehát egy fázisra fázisra az online algoritmus várható költsége H_k-szor az offline költség, amiből következik az állítás.