Amirreza Vishteh

HW2

26/8/1401

Operation System

سوال بخش عملی:

حل این سوال با استفاده از monte carlo tree search دارای مراحل زیر است:

ما یک کلاس داریم که board داده شده را تبدیل به object با ویژگی هایی مثل تعداد visit و value که همان تعداد دفعات منجربه برد است میکنیم همچنین ucb که از فرمول زیر محاسبه میشود نیز هست:

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\text{PARENT}(n))}{N(n)}}$$

- $N(n)$ = number of rollouts from node n
- $U(n)$ = total utility of rollouts (e.g., # wins) for Player(Parent(n))

1. selection:ob

در ابتدا نوبت ما است و مثلا یک خانه را مثل بالا انتخاب میکنیم

حال از اینجا الگوریتم شروع به کار میکند و در ابتدا با استفاده از ucb بین node های موجود یکی که دارای max(ucb) است را انتخاب میکند که در ابتدا فقط یک انتخاب وجود دارد مثل بالا ولی با اضافه شدن stateهای جدید باید از بین انها انتخاب انجام شود

2. **Expansion:**

در این مرحله درواقع childrens های ممکن برای این مورد select شده بر اساس پر کردن یکی از خانه های خالی ساخته میشود.

در اخر باید یک children برگداند این کار میتواند چند روش داشته باشد :

1. برگرداندن رندم از اعضا childrens

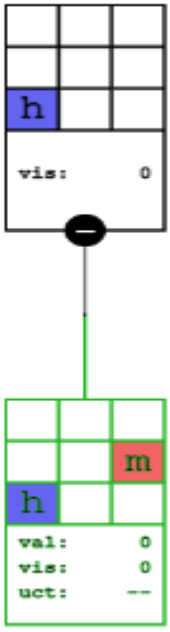
```
return random.choice(childrens)
```

2. رندوم برگرداندن براساس وزن که همان ucb است

```
    ucbs=[]
for i in childrens:
    calculateucb(i)
    ucbs.append(i.ucb)
if len(childrens)!=0:
    return random.choices(childrens,weights=ucbs,k=1)[0]
```

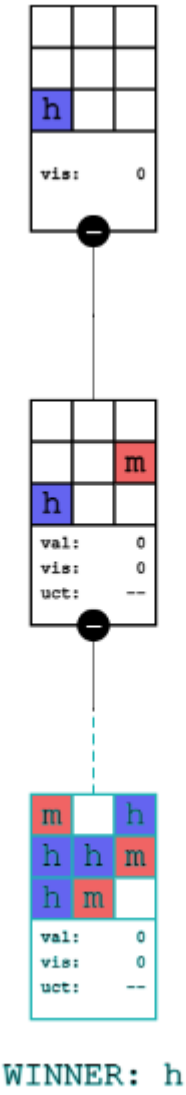
3. انتخاب با همان selection(root)

که از بین روش های بالا روش سوم از همه بهتر عمل کرد



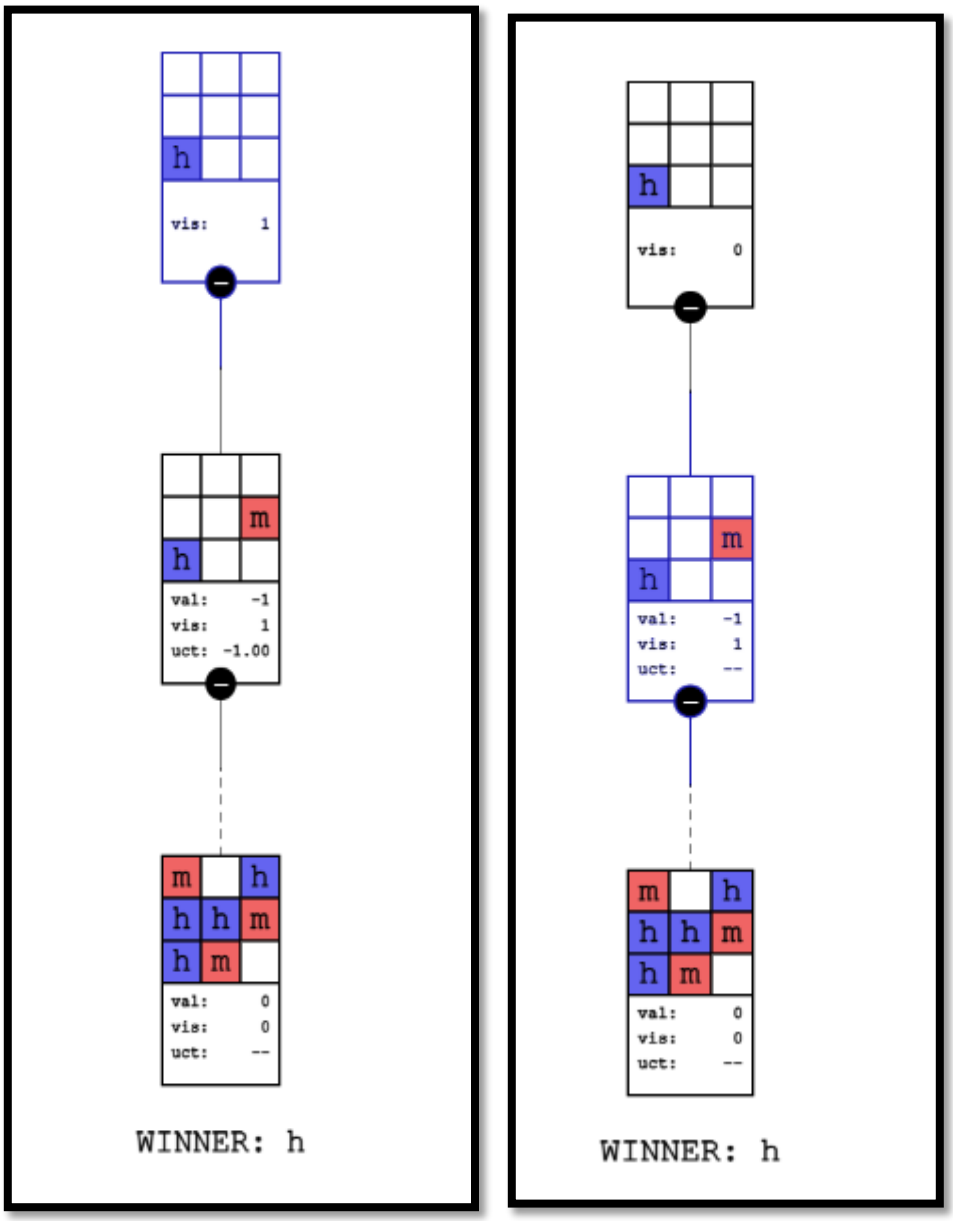
3.Simulation:

در این مرحله برنامه این فرزند انتخاب شده را به عنوان یک board تا اخر و رسیدن به نتیجه به شکل رندوم با خود کامپوتر بازی کند و یک score برگرداند که من با توجه به شرایط برای حالت های مساوی و برد برنامه و برد player امتیاز های بهینه را به دست اورده ام



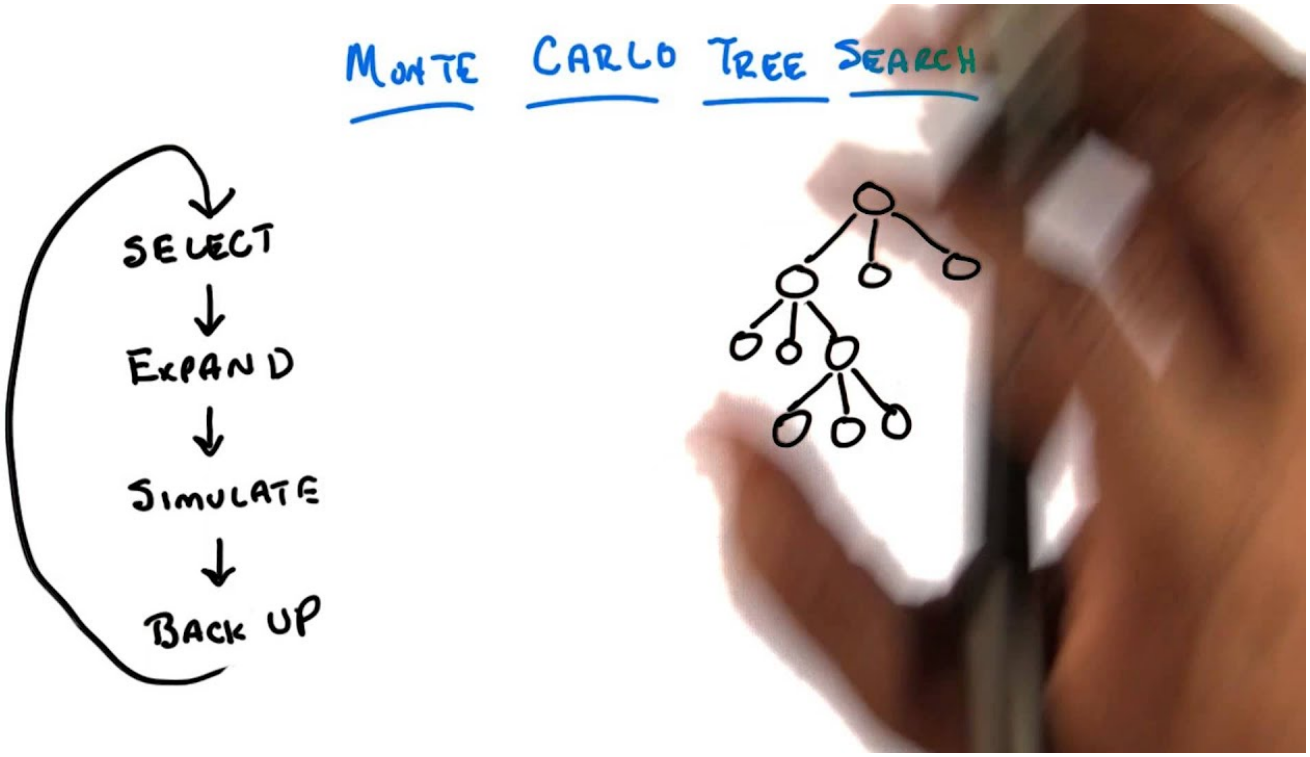
4. Backpropagation:

در این مرحله با از نتیجه به دست امده در مرحله قبل استفاده میکنیم و valueوvisit را برای node های parent به روز میکنیم



در اخر دوباره به مرحله اول باز گشته و کار را تکرار میکنیم به تعداد iteration ها که یک عدداز پیش تعیین شده است

در اخر اونی که ucb بزرگتری دارد از بین فرزندان حالت بهینه است و باید به ان رفت



توضیح سوال تئوری:

1. Model checking:

این روش با استفاده از جدول درستی است به این شکل که گزاره ها را برای فهمیدن اینکه درست اند یا نه از جدول چک میکنیم
برای تأیید صحت راه حل پیشنهادی برای یک مشکل استفاده می شود
در واقع زمانی که می خواهیم بفهمیم یک عبارت درست است یا نه یک راه این است که جدول درستی را رسم کنیم و ببینیم در چه حالت هایی KB(knowledge base) درست است در این خط ها ارزش بقیه گزاره هم معلوم میشود
درستی یا نادرستی عبارت هایی از جدول entails میشوند که در تمام موارد درستی KB دارای یک حالت اند یا در همه این حالات درست اند یا غلط

مثال:

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

2.theorem proving:

استفاده از قواعد منطقی به جای جدول درستی است در واقع با sequence از نتایج با استفاده از قواعد اثبات میشود.

در اینجا عبارتی که میخواهیم اثبات کنیم اگرA باشد انگاه اگر KB/\~A عبارت unsatisfaible شودانگاه عبارت درست است

در اثبات از قواعدی مانند Modus ponens و And Elimination و هم ارزی ها استفاده میکنیم

مثل:

P/\(p=>q)=p/\(~p\/q)=q