۱) عامل عکس العمل

بیشترین تاثیر و اهمیت را فاصله ی منهتنی کمتر از ۲ با روح دارد. در نتیجه مقدار منفی بی نهایت را بر می گردانیم. و در غیر این حالت فاصله پکمن با روح برایمان مهم نیست و در هر صورت به سمت غذا می رویم

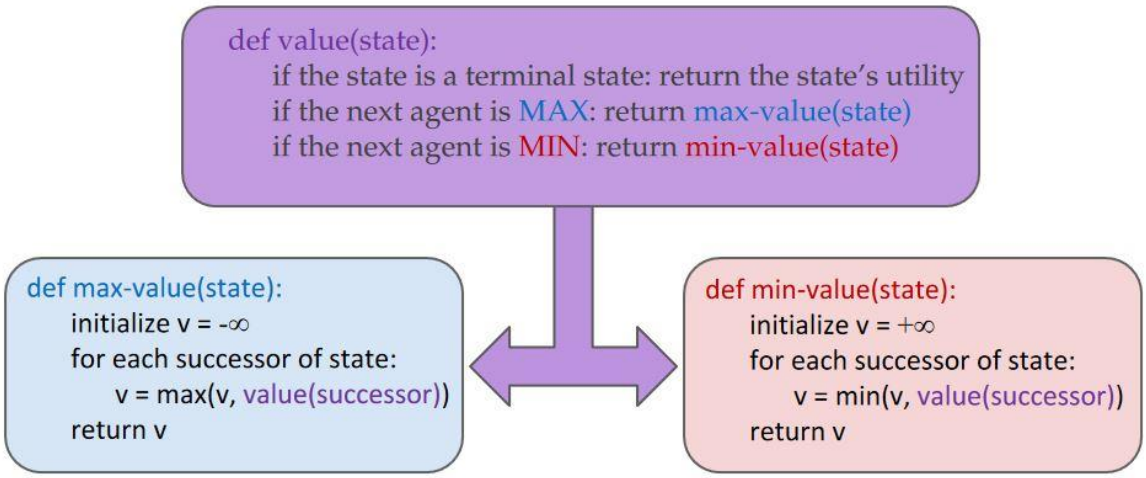
برای اینکه تاثیر فاصله ی غذای نزدیک تر بیشتر باشد، باید معکوس آن فاصله را نگه داریم در این صورت تاثیر غذای نزدیک تر بیشتر است (فاصله ی منهتنی) (و اگر غذایی نباشد به این معناست که همه ی غذا ها خورده شده و استیت بسیار خوبیست و مقدار ۲ را برای تاثیر غذا در نظر میگیریم (minFood = ۰,۵))

اختلاف امتیاز استیت جدید و قبلی را هم نیاز داریم (این اختلاف برایمان مهم است و می تواند مثبت یا منفی باشد)

زمان سفید بودن روح ها را هم با هم جمع میزنیم هرچه این مقدار بیشتر باشد استیت بهتری است

```
return (successorGameState.getScore() - currentGameState.getScore()) + ۱,۰ /
minFood + scaredTimes
```

۲) مینیماکس



پیاده سازی کلی به صورت بالاست اما ما در اینجا تابع valueی بالا را در min_value گنجانده ایم و همچنین در مورد عامل پکمن، نه تنها مقدار v را بر می گردانیم بلکه اکشنی که utility گره ماکس را بیشینه کرده را نیز لازم داریم زیرا عامل باید به آن جهت حرکت کند.

طبق گفته سوال ابتدا پکمن و سپس روح ها بازی میکنند. پس در ریشه درخت ما پکمن است و با تابع max_value شروع می کنیم و سپس روح های ما بازی می کنند (تابع min_value)

در هر عمق، ابتدا پکمن بازی میکند و هر کدام از روح ها یک بار. پس بعد از آخرین روح، برای حرکت پکمن به تابع max_value مقدار عمق+۱ را می دهیم (همچنین در ابتدای max_value چک می کنیم که آیا به عمق مورد نظر رسیده ایم یا چک می کنیم که آیا بازی تمام شده ، در این صورت مقداری که باید برگردانده شود باید از evaluationFunction گرفته شود)

برای پیاده سازی تابع max_value تمام حرکات مجاز از آن استیت را استخراج کنیم و با یک حلقه روی تمام ساکسسورها مقدار آنها را بدست بیاوریم. از آنجایی که بعد از پکمن نوبت روح هاست پس این ساکسسورها برای پکمن، مقدارشان با min_value بدست میاید

برای تابع min_value نیز به همین صورت است با این تفاوت که چک میکنیم نوبت بعدی نوبت پکمن است یا روح بعدی و اگر نوبت بعدی نوبت روح باشد، مقدار آن ساکسسور با تابع min_value به دست میاید اما اگر نوبت پکمن باشد با max_value

بررسی کنید :

در مینیماکس روح ها هم تخاصمی بازی می کنند بنابراین هر دو روح به سمت پکمن می روند و در هر حال آن را می گیرند.
بنابراین پکمن برای بیشینه کردن utility خود، دست به خودکشی میزند تا زمان و حرکت کمتری مصرف کند و امتیاز منفی کمتری بگیرد.

۳) هرس آلفا-بتا

α: MAX's best option on path to root
β: MIN's best option on path to root

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v
```

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

به توابع max_value و min_value مقدار آلفا و بتا را پاس می دهیم

پیاده سازی این بخش همانند بخش قبلی است و هرس را به صورت بالا انجام می دهیم (بدون در نظر گرفتن مساوی در if)

مقدار اولیه ی آلفا و بتا به ترتیب منفی بی نهایت و مثبت بی نهایت است

۴) مینیماکس احتمالی

پیاده سازی تابع max_value در این قسمت مشابه قسمت مینیماکس و برای محاسبه ی expected value باید مقدار گره های فرزند را در احتمالشان ضرب کرده و با هم جمع کنیم (در اینجا از آنجایی که احتمال های برابر دارند، احتمال هر کدام می شود ۱/تعداد حرکات مجاز (گره ها))

در تابع exp_value ، اکشنی را برنمی گردانیم زیرا اکشن ها برای روح ها احتمالی هستند و روح ها لزوما به این جهت حرکت نمیکنند

بررسی کنید :

python pacman.py -p AlphaBetaAgent -l trappedClassic -a depth=۳ -q -n ۱۰

با اجرای این خط، پکمن تمام ۱۰ بازی را می بازد (با هرس آلفا بتا)

python pacman.py -p ExpectimaxAgent -l trappedClassic -a depth=۳ -q -n ۱۰

و با اجرای این خط می بینیم که پکمن ۷ بازی از ۱۰ تا را برده است (با Expectimax)

```
C:\Users\sug\Desktop\multiagents>python pacman.py -p ExpectimaxAgent -l trappedClassic -a depth=3 -q -n 10
Pacman emerges victorious! Score: 532
Pacman died! Score: -502
Pacman died! Score: -502
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 532
Pacman emerges victorious! Score: 532
Pacman died! Score: -502
Average Score: 221.8
Scores:        532.0, -502.0, -502.0, 532.0, 532.0, 532.0, 532.0, 532.0, 532.0, -502.0
Win Rate:      7/10 (0.70)
Record:        Win, Loss, Loss, Win, Win, Win, Win, Win, Win, Loss
```

به این دلیل که در این حالت روح ها (حریف) تخاصمی بازی نمی کنند بلکه احتمالی باز می کنند در نتیجه پکمن ما به طور میانگین نیمی از بازی ها را می برد

۵) تابع ارزیابی

در این سوال بر اساس استیت فعلی تصمیم گرفته ایم

برای اینکه تاثیر فاصله ی غذای نزدیک تر بیشتر باشد، باید معکوس آن فاصله را نگه داریم در این صورت تاثیر غذای نزدیک تر بیشتر است (فاصله ی منهتنی) (و اگر غذایی نباشد به این معناست که همه ی غذا ها خورده شده و استیت بسیار خوبیست و مقدار ۲ را برای تاثیر غذا در نظر میگیریم (minFood = ۰٫۵))

همچنین امتیاز استیت فعلی را هم در نظر می گیریم

در نتیجه داریم:

```
return ۱,۰ / closestFoodDist + score
```