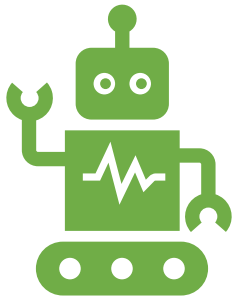
Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Информирано пребарување



Аудиториски вежби по курсот
Вештачка интелигенција

2023/2024

Проблем на пребарување низ простор на состојби

- Проблем на пребарување низ простор на состојби се состои од:
  - Простор на состојби – множество на сите можни состојби.
  - Почетна состојба – состојба од која започнува пребарувањето
  - Проверка за постигнување на цел – функција која проверува дали моменталната состојба е цел.
- Решение на проблемот на пребарување низ простор на состојби е секвенца на акции, наречена план, која ја трансформира почетната состојба во целна состојба.
- Овој план се креира со помош на алгоритми на пребарување.

Формулација на проблем како пребарување низ простор на состојби

- Дефинираме состојба според природата на проблемот
- Дефинираме почетна и целна состојба
- Дефинираме акции/оператори кои ја менуваат состојбата
- Дефинираме дозволени (легални) состојби за проблемот

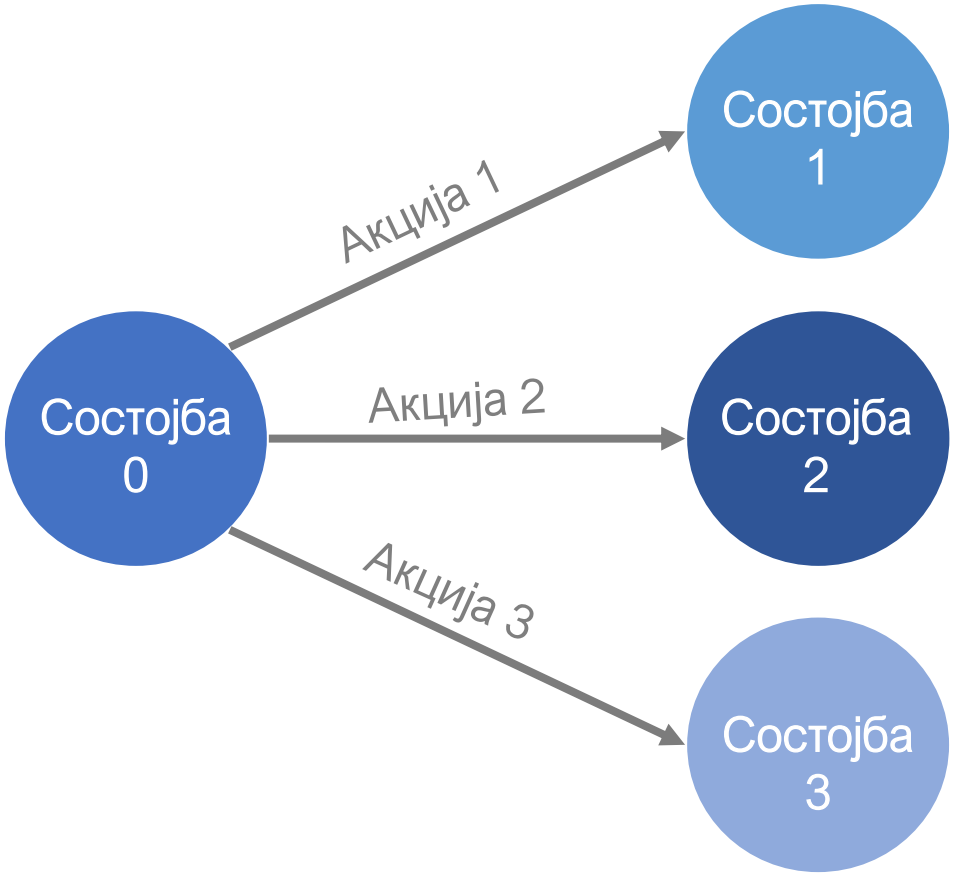
Состојба во Python

- Во Python, состојбата може да се претстави на различни начини
- Еден начин е да ја дефинираме како торка од нејзините атрибутски вредности
- Пример:
  - Позицијата на објект на мапа можеме да ја определиме според географската широчина и географската височина на која се наоѓа
  - Позицијата на објект кој се наоѓа во Деканатот на ФИНКИ би била:
    - (42.004113, 21.409549)

Акција/Оператор

- Акцијата влијае на состојбата со тоа што ги менува атрибутските вредности
- На пример, доколку објектот претставува летало кое лета над ФИНКИ и се движи со брзина 1km/s на север (акцијата му е движење на север со брзина 1km/s), следната состојба или позиција на објектот ќе биде (после една секунда):
    - (42.014119, 21.409528)
- Имаме движење или промена на состојбата од (42.004113, 21.409549) во (42.014119, 21.409528)

Состојби-Акции-Состојби

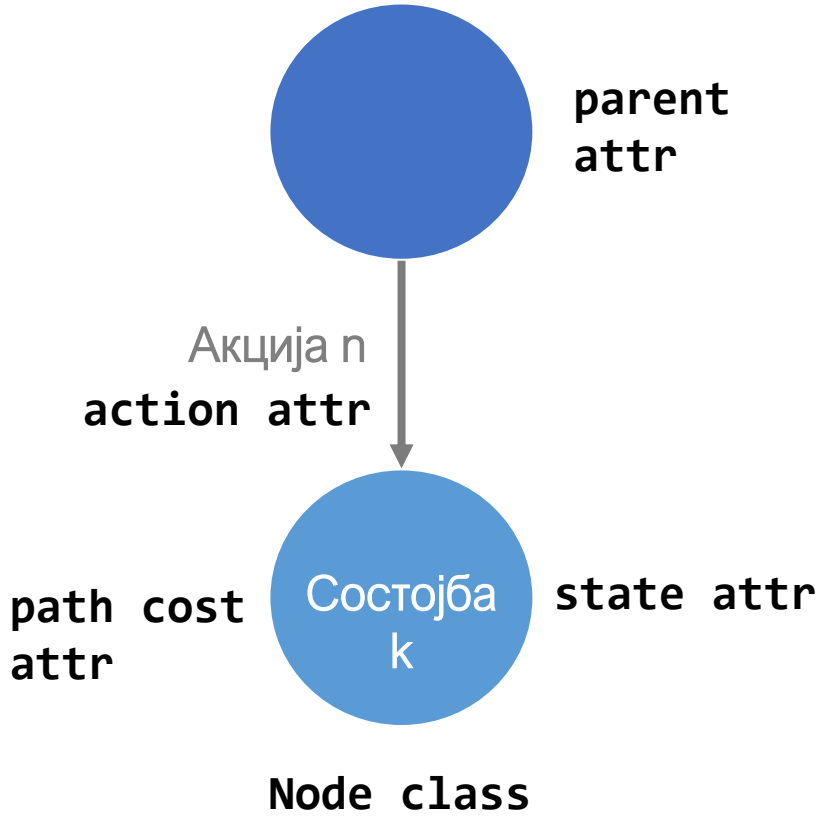


Дефинирање на проблем во Python

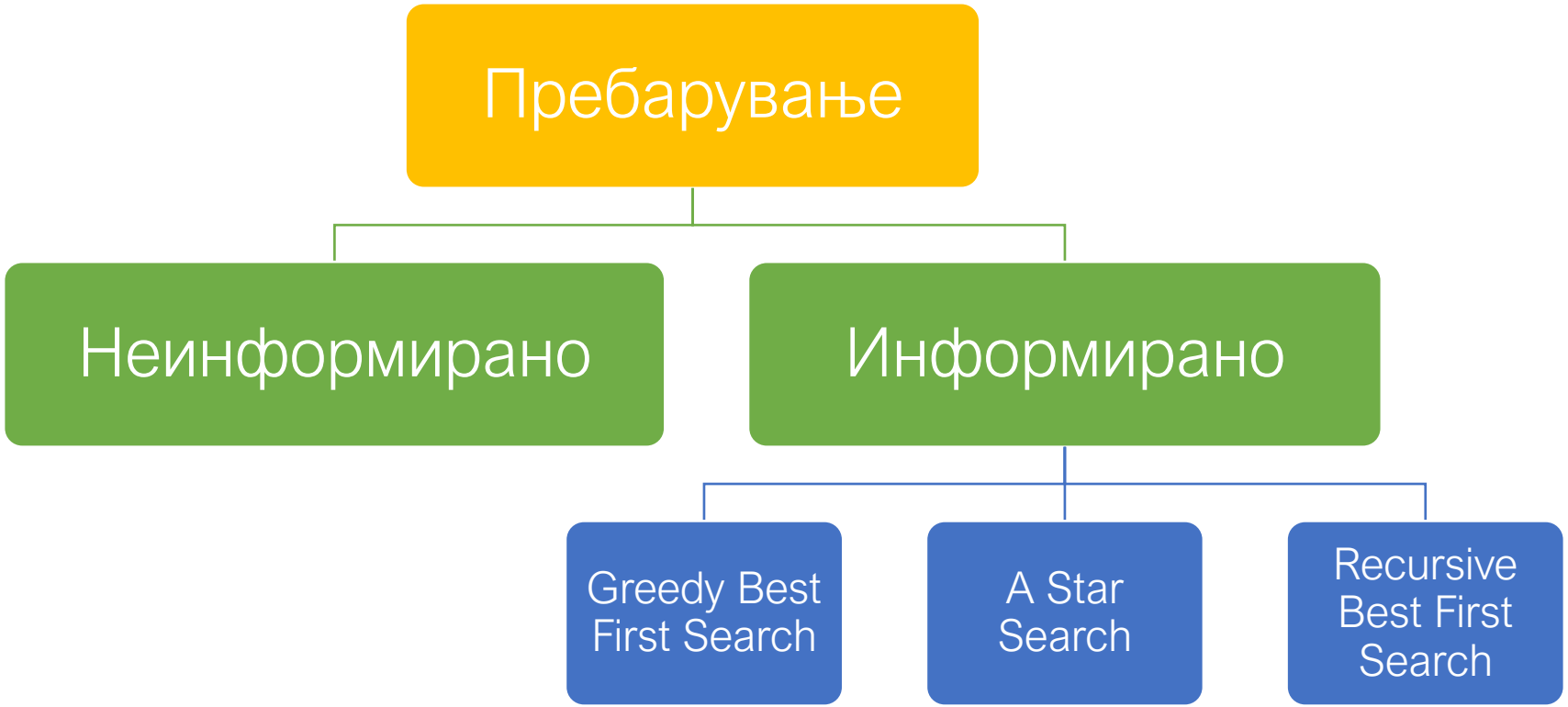
- Дефинираме класа за структурата на проблемот кој ќе го решаваме со пребарување
- Класата Problem е апстрактна класа од која правиме наследување за дефинирање на основните карактеристики на секој еден проблем што сакаме да го решиме

Дефинирање на јазел во Python

- Дефинираме класа Node за структурата на јазел од пребарувачко дрво
- Секој јазел содржи:
  - Покажувач кон родителот (parent)
  - Покажувач кон состојбата претставена со јазелот (state)
  - Акција со која состојбата претставена со јазелот е добиена од состојбата - родител (action)
  - Цена на патот од почетниот јазел до тој јазел
- Класата Node не се наследува



Алгоритми на информирано пребарување



Алгоритми на информирано пребарување (2)

- Секој од алгоритмите за пребарување ќе има:
  - Граф проблем, кој го содржи почетниот јазел S и целниот јазел G.
  - Стратегија, со која се опишува начинот на кој графот ќе се изминува за да се стигне до G.
  - Хевристичка функција која служи да го извести пребарувањето за насоката кон целта, односно овозможува информиран начин да се претпостави кој соседен јазел ќе доведе до целта поефективно.
  - Податочна структура за да се сочуваат сите возможни состојби (јазли) до кои може да се стигне од моменталните состојби.
  - Дрво, кое се добива преку изминувањето до целниот јазел.
  - План за решението, што е патека, односно секвенца на јазли од S до G.

Best first search

- Best first search претставува пребарување кое одлучува кој јазел треба да се посети следно со проверка кој јазол е најперспективен и потоа се посетува најперспективниот јазел. За ова се користи функција за проценка за да се одлучи на погодноста, **f(n)**.
- Greedy best first search е best first пребарување каде што се специфицира дека **f(n) = h(n)**.
- A star search (A* search) е best first пребарување каде што се специфицира дека **f(n) = g(n) + h(n)**.
- Recursive best first search ја ограничува рекурзијата преку следење на f-вредноста на најдобриот алтернативен пат од било кој јазел предок (еден чекор гледање нанапред). Recursive best first search претставува верзија на A* пребарувањето кое користи ограничена меморија.

Хевристичка функција

- A* сигурно може да ја пронајде најкратката патека што води од дадениот почетен јазел до кој било јазол на целта во проблем графот, ако хевристичката функција **h(n)** е прифатлива.
- Односно важи дека $h(n) \leq h^*(n)$, за сите јазли **n**, каде што **h*(n)** е вистинската вредност.

Проблем: Сложувалка

- Дадена е сложувалка 3x3, на која има полиња со броеви од 1 до 8 и едно празно поле. Празното поле е обележано со '*'.

*	3	2
4	1	5
6	7	8

- Проблемот е како да се стигне од некоја почетна распределба на полињата до некоја посакувана, на пример:

*	1	2
3	4	5
6	7	8

Проблем: Сложувалка (2)

- Акции: акциите ќе ги разгледуваме како придвижување на празното поле, па можни акции се:
  - Горе
  - Долу
  - Лево
  - Десно
- При дефинирањето на акциите, мора да се внимава дали тие воопшто можат да се преземат во дадената сложувалка.

Дефиниција на состојба

- Состојбата ќе ја дефинираме како стринг кој ќе има 9 знаци (по еден за секое бројче, плус ‘*’).
- Притоа, стрингот ќе се пополнува со изминување на сложувалката од првиот кон третиот ред, од лево кон десно.
- Пример: состојбата за почетната сложувалка е „*32415678“, а за финалната сложувалка е „*12345678“.

Примери за хевристика

1. Број на полиња кои не се на вистинското место
2. Менхетен растојание до целната состојба

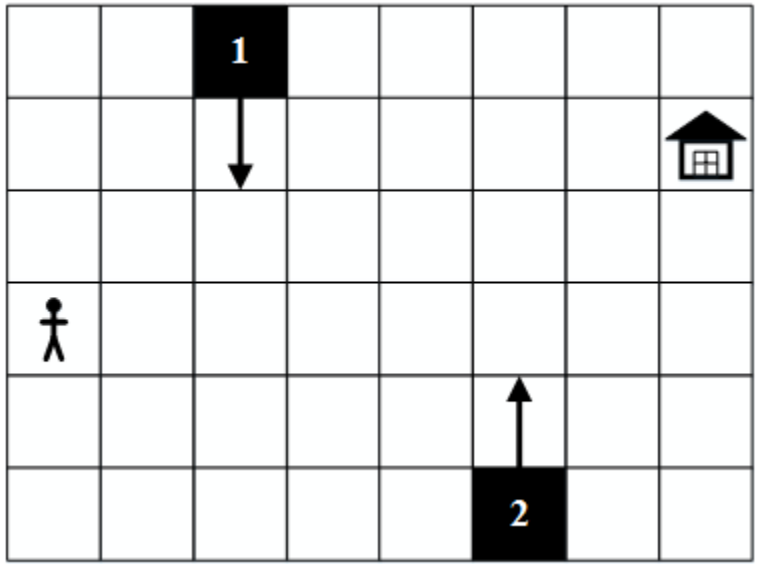
Менхетен растојание

- За да можеме да го дефинираме растојанието, потребно е да дефинираме координатен систем
- Почетокот на координатниот систем е поставен во долниот лев агол на сложувалката
- Дефинираме речник за координатите на секое поле од сложувалката.
- Дефинираме функција која пресметува Менхетн растојание за сложувалката.
- Оваа функција на влез прима два цели броеви, кои одговараат на две полиња на кои се наоѓаат броевите за кои треба да пресметаме растојание

0,2	1,2	2,2
0,1	1,1	2,1
0,0	1,0	2,0

Задача 1: Истражувач

- Предложете соодветна репрезентација и напишете ги потребните функции во Python за да се реши следниот проблем за кој една можна почетна состојба е прикажана на сликата

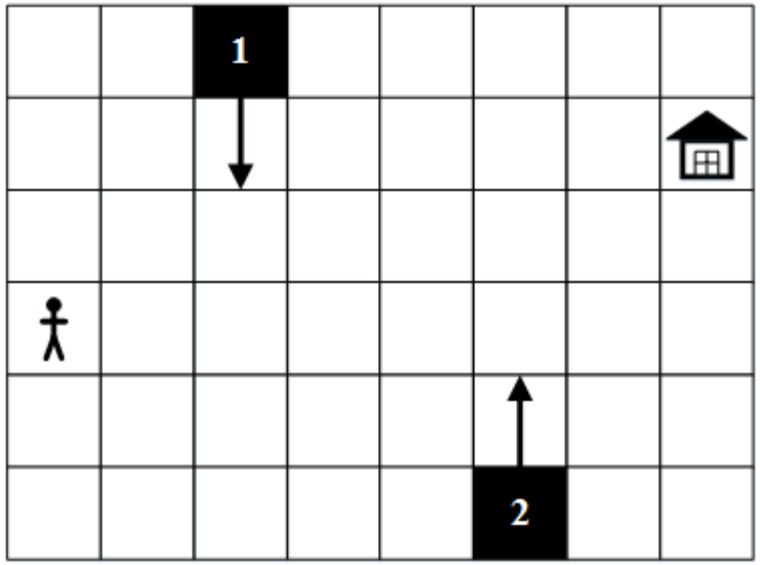


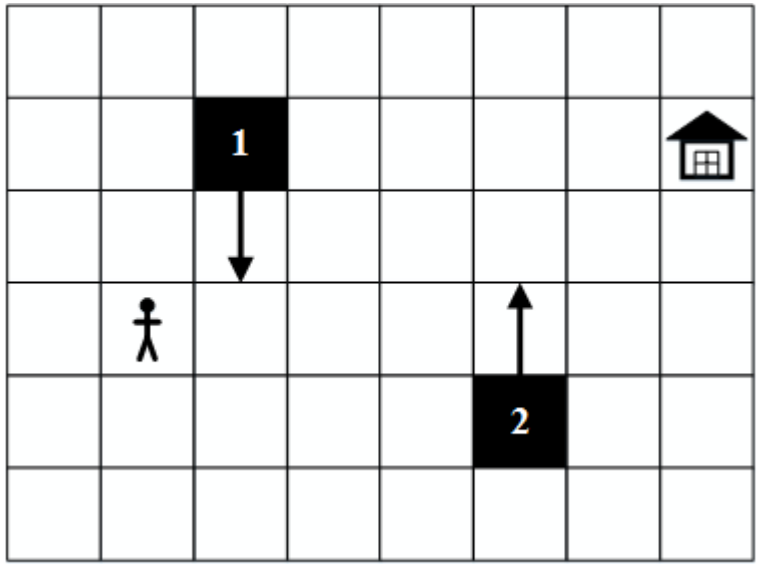
Задача 1 (2)

- Потребно е човечето безбедно да дојде до куќичката. Човечето може да се придвижува на кое било соседно поле хоризонтално или вертикално.
- Пречките 1 и 2 се подвижни, при што и двете пречки се движат вертикално. Секоја од пречките се придвижува за едно поле во соодветниот правец и насока со секое придвижување на човечето.

Задача 1 (3)

- Притоа, пречката 1 на почетокот се движи надолу, додека пречката 2 на почетокот се движи нагоре. Пример за положбата на пречките после едно придвижување на човечето надесно е прикажан на десната слика.





Задача 1 (4)

- Кога некоја пречка ќе дојде до крајот на таблата при што повеќе не може да се движи во насоката во која се движела, го менува движењето во спротивната насока.
- Доколку човечето и која било од пречките се најдат на исто поле човечето ќе биде уништено.
- За сите тест примери изгледот и големината на таблата се исти како на примерот даден на сликите. За сите тест примери почетните положби, правец и насока на движење за препреките се исти. За секој тест пример почетната позиција на човечето се менува, а исто така се менува и позицијата на куќичката.
- Во рамки на почетниот код даден за задачата се вчитуваат влезните аргументи за секој тест пример.

Задача 1 (5)

- Движењата на човечето потребно е да ги именувате на следниот начин:
  - Right - за придвижување на човечето за едно поле надесно
  - Left - за придвижување на човечето за едно поле налево
  - Up - за придвижување на човечето за едно поле нагоре
  - Down - за придвижување на човечето за едно поле надолу

Задача 1 (6)

- Вашиот код треба да има само еден повик на функција за приказ на стандарден излез (print) со кој ќе ја вратите секвенцата на движења која човечето треба да ја направи за да може од својата почетна позиција да стигне до позицијата на куќичката.
- Треба да примените информирано пребарување. За имплементација на хевристичката функција треба да користите Менхетен растојание.

Задача 1 – Анализа на проблемот

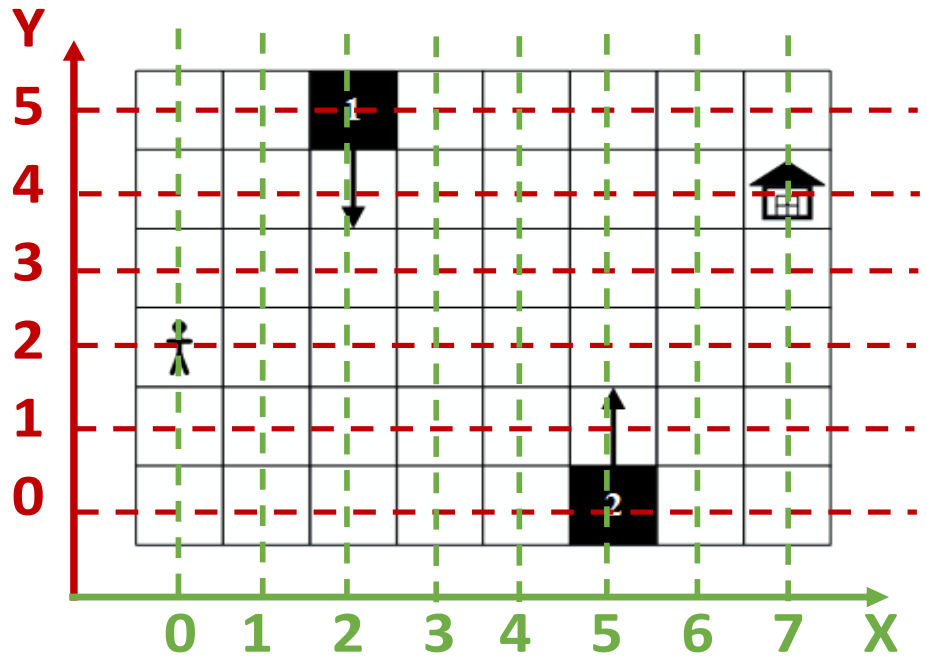
- Првиот чекор во анализата е дефинирањето на состојбата
- Без оглед на проблемот, состојбата мора да содржи променливи преку кои ќе се следат промените кои се случуваат како резултат на акциите (операциите) кои се извршуваат
- Начинот на структурирање на состојбата е личен избор на програмерот
  - Секогаш треба да се размислува колку „лесно“ може да се манипулира со структурата која ја дефинира состојбата т.е. како ќе се имплементираат функциите кои тековната состојба ќе ја трансформираат во следна

Задача 1 – Анализа на проблемот (2)

- Во конкретниот проблем најпрво треба да се идентификува што е тоа што се менува при преземање на одредена акција
  - Акциите се всушност потезите кои се дефинирани и со кои може да се „смени“ проблемот, а со цел да се доведе до некаква целна состојба
    - Right, Left, Up, Down
  - Човечето се придвижува како резултат на акциите и ја менува позицијата, па треба да обезбедиме начин да го следиме
  - Пречките се движат самостојно, но сепак се придвижуваат само кога и човечето се движи
    - Во општ случај не треба да ја следиме позицијата на пречките бидејќи не зависи од акциите кои се преземаат
    - Во конкретниот проблем поради значително поедноставување на имплементацијата во состојбата ќе ја додадеме и оваа информација
      - Ваквото поедноставување може да се искористи во било кое сценарио каде промените кои НЕ СЕ последица на акциите се случуваат синхронизирано со промените кои СЕ последица на акциите

Задача 1 – Анализа на проблемот (3)

- Во проблеми кај кои има движење на табла наједноставен пристап за „лоцирање“ на различните елементи е да се постават координатни оски, при што секое поле од таблата ќе биде идентификувано преку уникатна комбинација на неговите координати
- За поставениот проблем таквата претстава е дадена на сликата



Задача 1 – Анализа на проблемот (4)

- Со така поставените оски можеме формално да ги дефинираме формалните елементи на пребарувањето преку користење на X и Y координатите
- Почетна состојба:
  - Во општ случај (без пречки), во состојбата треба да ги чуваме само координатите на човечето, па ќе имаме торка (X,Y) или за дадената состојба на претходната слика (0,2)
  - Во поедноставената имплементација во состојбата треба да се води информација и за пречките (нивната позиција и моментална насока на движење), па ќе имаме торка (Xc, Yc, (X1, Y1, N1), (X2, Y2, N2)), или за дадената состојба од претходната слика (0, 2, (2, 5, -1), (5, 0, 1))
    - Движењето надолу ја намалува Y координатата, па затоа е претставено со -1. Аналогно, нагоре ја зголемува, па е претставено со 1
- Целна состојба:
  - Во општ случај човечето треба да дојде на позиција на куќичката, што значи дека целта е експлицитно дефинирана. Во конкретниот пример со (7,4)
  - Во поедноставената имплементација единствено битно е човечето да стигне на позиција на куќичката, додека пречките можат да бидат произволни. Значи има имплицитна цел па ќе треба истата соодветно да се дефинира преку функција која единствено ќе го проверува човечето

Задача 1 – Анализа на проблемот (5)

- Акции
  - Во општ случај акцијата треба да ја промени позицијата (координатите) на човечето во зависност од акцијата. Од аспект на имплементација тоа значи едноставно промена на една од вредностите во торката.
    - На пример, ако акцијата е Gore трансформацијата која треба да ја направиме е

$$(X,Y) \rightarrow (X,Y+1)$$

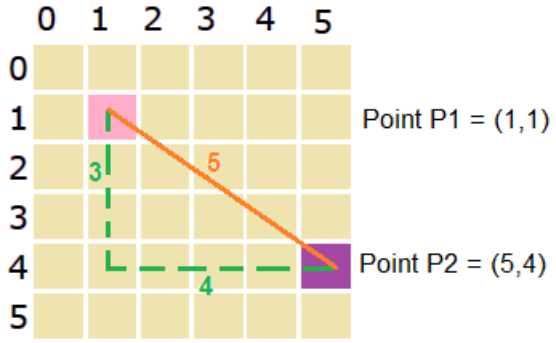
  - Во поедноставениот случај покрај позицијата на човечето треба да ја ажурираме и позицијата на пречките (истото секако треба да се направи и во општиот случај, но надвор од имплементацијата за акциите). Треба да се направат посебни проверки за тоа кога пречките стигаат до крајот на таблата и соодветно да им се смени насоката на движење
  - И во двата случаи треба да се направи проверка за тоа дали новата состојба е легална и да се изгенерираат само легалните следни состојби
    - Човечето не смее да биде на иста позиција со некоја од пречките
    - Човечето не смее да излезе од границите на таблата

Задача 1 – Анализа на проблемот (6)

- Хевристика
  - Овој дел е единствениот различен во однос на анализата на проблемот при решавање со неинформирано пребарување
  - Менхетен (Manhattan) растојанието се пресметува како збир од апсолутни разлики помеѓу димензиите
    - Во правоаголен координатен систем тоа значи збир помеѓу апсолутните разлики на X координатите и Y координатите на две точки
    - $ManDist(A,B)=|X_A – X_B| + |Y_A – Y_B|$

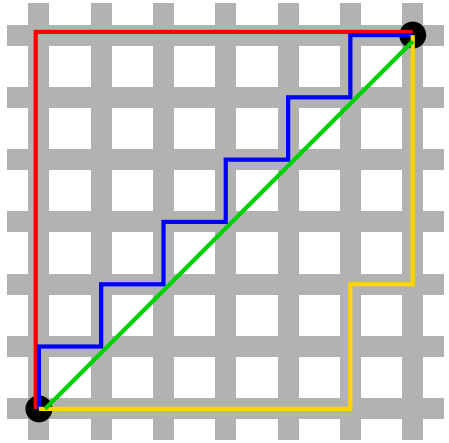
Задача 1 – Анализа на проблемот (7)

- Во проблеми со правоаголна табла Менхетен растојанието е корисно бидејќи најчесто одговара на бројот на чекори кои се прават
  - На сликата долу со зелено е означено Евклидовото (квадратно) растојание
  - Со останатите бои се означени можни патеки за да се стигне од една до друга точка
  - Сите патеки имаат иста должина која е еднаква на Менхетен растојание



Euclidean distance = $\sqrt{(5-1)^2 + (4-1)^2} = 5$

Manhattan distance = $|5-1| + |4-1| = 7$

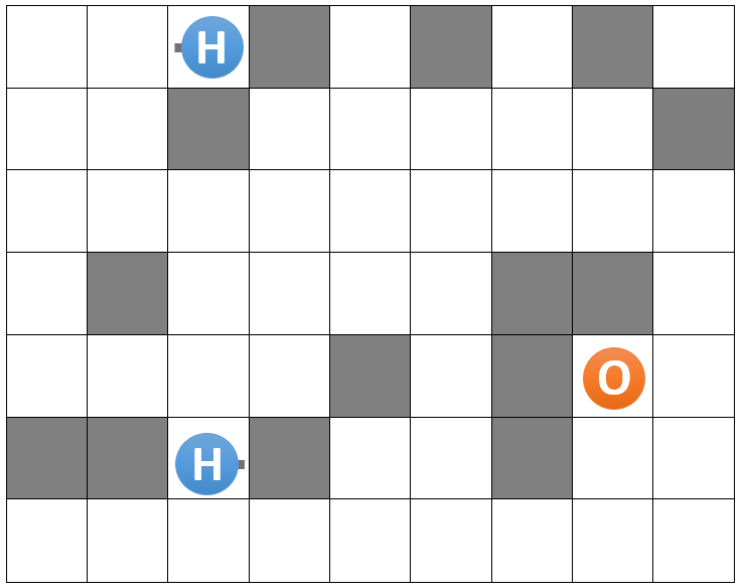


Задача 2: Молекула

- Предложете соодветна репрезентација и напишете ги потребните функции во Python за да се реши следниот проблем за кој една можна почетна состојба е прикажана на сликата на следниот слајд.
- На табла 7x9 поставени се три атоми (внимавајте, двата H-атоми се различни: едниот има линк во десно, а другиот има линк во лево). Полињата обоени во сива боја претставуваат препреки.

Задача 2 (2)

- Играчот може да ја започне играта со избирање на кој било од трите атоми.
- Играчот во секој момент произволно избира точно еден од трите атоми и го „турнува“ тој атом во една од четирите насоки: горе, долу, лево или десно.

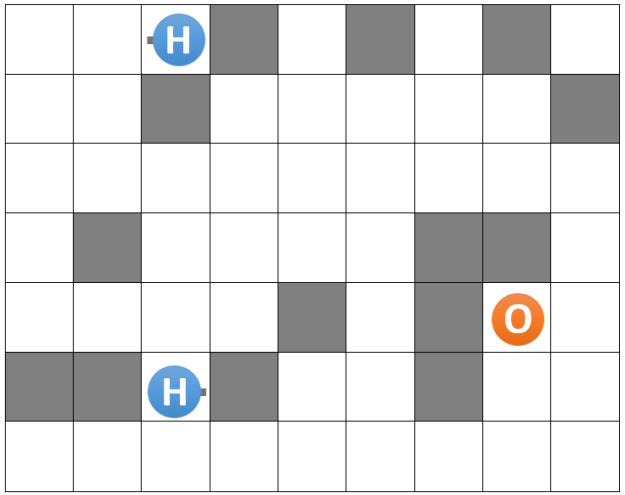


Задача 2 (3)

- Движењето на „турнатиот“ атом продолжува во избраната насока се’ додека атомот не „удри“ во препрека или во некој друг атом (атомот секогаш застанува на првото поле што е соседно на препрека или на друг атом во соодветната насока).
- Не е возможно ротирање на атомите (линковите на атомите секогаш ќе бидат поставени како што се на почетокот на играта). Исто така, не е дозволено атомите да излегуваат од таблата.

Задача 2 (4)

- Целта на играта е атомите да се доведат во позиција во која ја формираат „молекулата“ прикажана десно од таблата. Играта завршува во моментот кога трите атоми ќе бидат поставени во бараната позиција, во произволни три соседни полиња од таблата.
- Потребно е проблемот да се реши во најмал број на потези.



Задача 2 (5)

- За сите тест примери изгледот и големината на таблата се исти како на примерот даден на сликата. За сите тест примери положбите на препреките се исти. За секој тест пример се менуваат почетните позиции на сите три атоми, соодветно.
- Во рамки на почетниот код даден за задачата се вчитуваат влезните аргументи за секој тест пример.

Задача 2 (6)

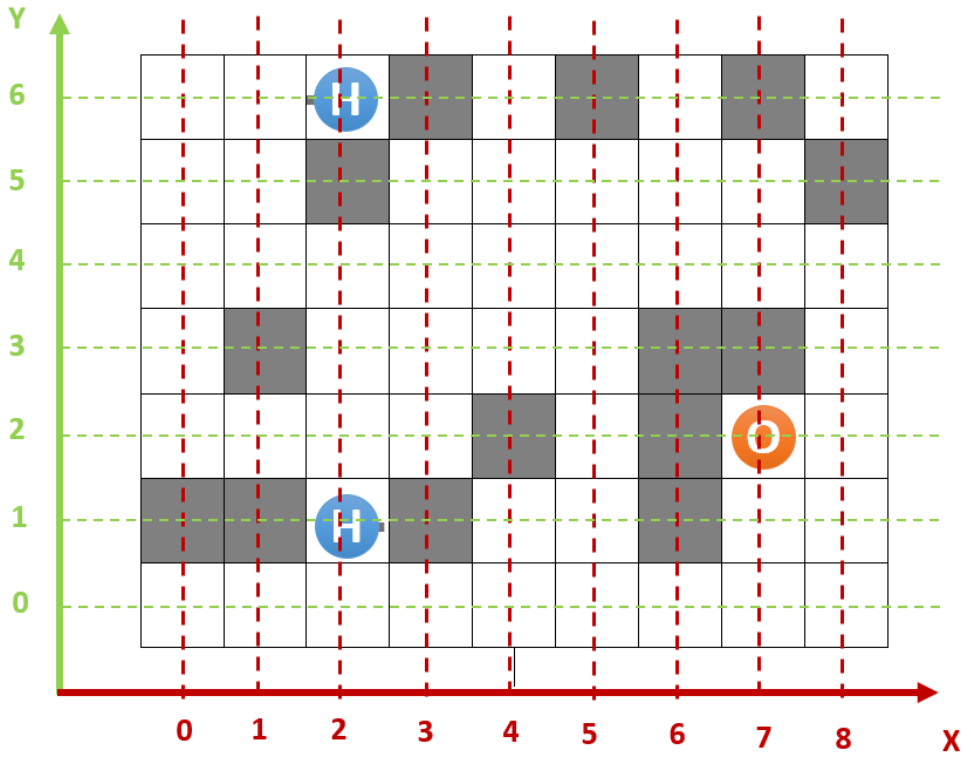
- Движењата на атомите потребно е да ги именувате на следниот начин:
  - RightX - за придвижување на атомот X надесно (X може да биде H1, O или H2)
  - LeftX - за придвижување на атомот X налево (X може да биде H1, O или H2)
  - UpX - за придвижување на атомот X нагоре (X може да биде H1, O или H2)
  - DownX - за придвижување на атомот X надолу (X може да биде H1, O или H2)

Задача 2 (7)

- Вашиот код треба да има само еден повик на функција за приказ на стандарден излез (print) со кој ќе ја вратите секвенцата на движења која треба да се направи за да може атомите од почетната позиција да се доведат до бараната позиција.
- Треба да примените информирано пребарување. Одредете прифатлива хевристичка функција.

Задача 2 – Анализа на проблемот

- Како и во претходниот проблем, се поставуваат координатни оски за да се дефинира позицијата на елементите кои треба да ги следиме



Задача 2 – Анализа на проблемот (2)

- Во овој проблем пречките се статични, па треба да се дефинираат како „глобални“ во рамки на имплементацијата
- Промените во проблемот настануваат како резултат на придвижување на молекулите, па нивните позиции ќе бидат дел од состојбата. Состојбата ќе биде торка од шест вредности за секоја од координатите на трите молекули: $(X_{H1}, Y_{H1}, X_O, Y_O, X_{H2}, Y_{H2})$

Задача 2 – Анализа на проблемот (3)

- Почетна состојба
  - За конкретниот пример, почетната состојба ќе биде (2, 1, 7, 2, 2, 6)
- Целна состојба
  - Целната состојба е имплицитна. Не е битна конкретната позиција на молекулите, туку нивната меѓусебна релативна позиција. Треба да се направи функција која ќе проверува дали атомите H1, O, H2 се позиционирани еден до друг и точно во тој редослед

Задача 2 – Анализа на проблемот (4)

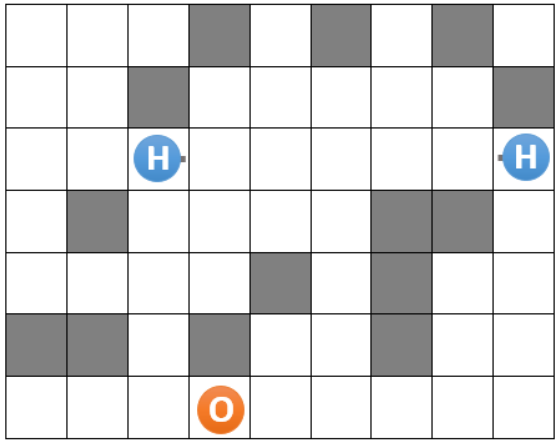
- Акции
  - Треба да се дефинираат посебни акции за движење на секој од атомите во секоја од четирите насоки
  - Бидејќи станува збор за “турнување” на атомите, движењето можеме да го гледаме како повеќекратно единечно поместување на атомите
    - Циклус во кој во секоја итерација има единечно поместување во соодветната насока, со услов за крај дефиниран или преку доаѓање до крај на табла или доаѓање до препрека или до друг атом

Задача 2 – Анализа на проблемот (4)

- Хевристика
  - Идеја 1: Збир на Менхетен растојанијата помеѓу молекулите H1 и O, и O и H2.
  - Идеја 2: Хевристичка функција која за секоја состојба пресметува минимален број на турнувања на паровите од атоми за тие да се спојат.

Задача 2 – Анализа на проблемот (5)

- Идеја 1: Збир на Менхетен растојанијата помеѓу молекулите H1 и O, и O и H2.
- ПРОБЛЕМ: Неприфатлива хевристика. Хевристиката враќа поголема вредност од вистинската вредност. Пример:



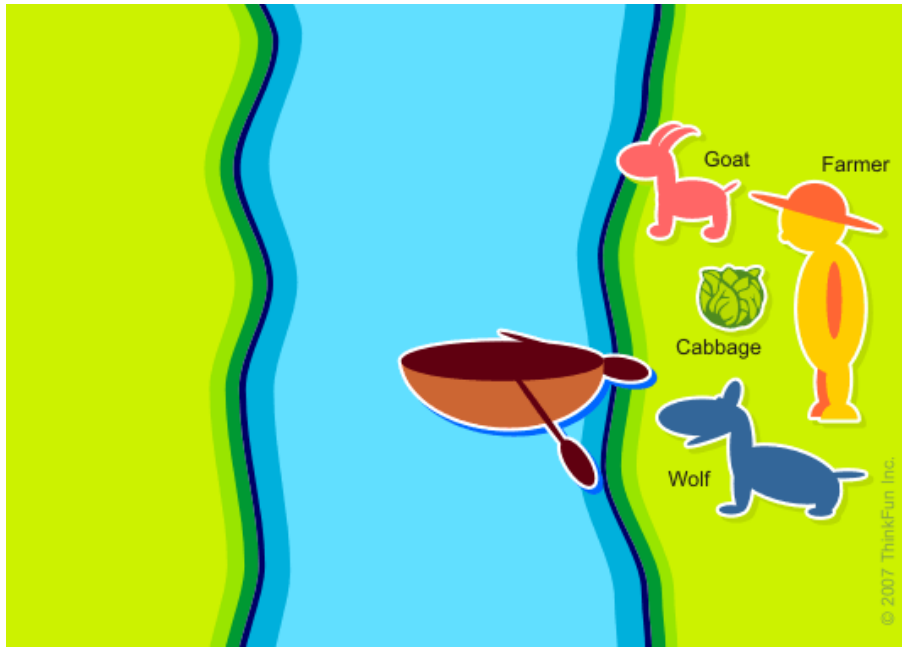
- Хевристиката за дадената состојба на сликата пресметува вредност 12, односно 4 (Менхетен растојанието помеѓу H1 и O) + 8 (растојанието меѓу O и H2).
- Вистинската вредност е всушност 3.

Задача 2 – Анализа на проблемот (6)

- Идеја 2: Хевристичка функција која за секоја состојба пресметува минимален број на турнувања на паровите од атоми за тие да се спојат.
  - Потребниот број на чекори за да се спојат H1 и O + Потребниот број на чекори за да се спојат O и H2.
  - Потребен број на чекори за да се спојат H1 и O
    - Ако H1 и O се во различни редици и H1 не е во левата колона веднаш до O, тогаш потребни се минимум 2 турнувања на H1;
    - Ако H1 и O се во различни редици и H1 е во левата колона веднаш до O, тогаш потребни се минимум 1 турнување на H1;
    - Ако H1 и O се во исти редици и H1 не е од левата страна на O, тогаш потребни се минимум 3 турнувања на H1;
    - Ако H1 и O се во исти редици и H1 е од левата страна на O, тогаш потребни се минимум 1 турнување на H1;
  - Потребен број на чекори за да се спојат O и H2: истата логика само проверка на позицијата на H2 од десно.
  - Специјален случај: Ако H1 и H2 се во иста редица на соодветната страна, а атомот O во друга редица, потребно е да се турне само атомот O, односно со претходната логика се пресметуваат 2 турнувања на H атомите на H атомите нагоре/надолу, а решение може да се добие само со 1 турнување на O.
- Во дадени случаи, хевристичката функција дава помала вредност од вистинската, но останува прифатлива функција.

Задача 3: Проблем на фармерот

- Предложете соодветна репрезентација и напишете ги потребните функции во Python за да се реши следниот проблем за кој почетната состојба е прикажана на сликата.



Задача 3 (2)

- Потребно е да се пренесат зелката, јарето, волкот и фармерот од источната страна на западната страна на реката
- Само фармерот го вози чамецот
- Во чамецот има простор за двајца патници: фармерот и уште еден патник
- Ограничувања:
  - Доколку останат сами (без присуство на фармерот):
    - **Јарето ја јаде зелката**
    - **Волкот го јаде јарето**

Задача 3 (3)

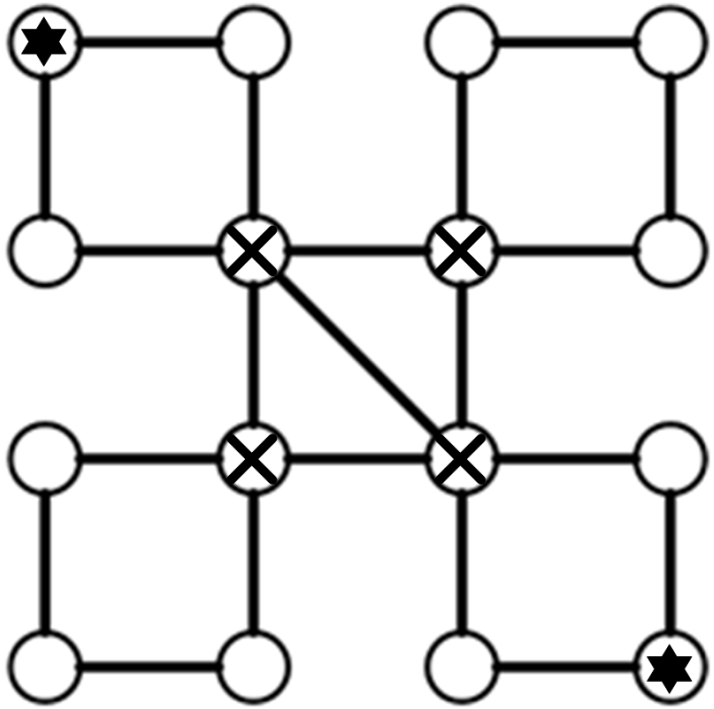
- Вашиот код треба да има само еден повик на функција за приказ на стандарден излез (print) со кој ќе ја вратите секвенцата од позиции на актерите која одговара на секвенцата на движења со која сите актери ќе бидат пренесени на западната страна на реката.
- Треба да примените информирано пребарување. Дефинирајте соодветна хевристика која ќе биде прифатлива за проблемот.

Задача 3 – Анализа на проблемот

- Состојбата е торка од позициите на фармерот, волкот, јарето и зелката, точно во тој редослед.
- Пример:
  - (“w”, “w”, “w”, “w”) значи дека сите се наоѓаат на западната страна (целна состојба)
  - (“e”, “e”, “e”, “e”) значи дека сите се наоѓаат на источната страна (почетна состојба)
- Акциите ќе се именуваат Farmer_nosi_X, каде X може да биде било кој од актерите во проблемот (ако X е фармерот тоа значи дека сам се вози во чамецот).
- Треба да се внимава фармерот и X да се наоѓаат на ист брег.
- Да не се наруши ниту едно од ограничувањата на проблемот.
- Целната состојба е експлицитна и е дадена погоре како пример за состојба.
- Хевристиката ќе ја моделираме како Хамингово растојание помеѓу состојби (број на различни букви помеѓу тековна и целна состојба).

Задача 4: Истражувач во граф

- На сликата е дадена почетна состојба за игра чие решавање треба да го дефинирате како пребарување низ простор на состојби.



Задача 4 (2)

- Играта е дефинирана на следниот начин: Играчот може да ја започне играта со позиционирање на било кое од крукчињата означени со X. Играчот може да се придвижи од крукчето на кое тековно се наоѓа до некое друго крукче ако постои врска помеѓу нив. Играчот собира ѕвезда ако застане на крукче на кое ѕвездата се наоѓа. Целта на играчот е да ги собере сите ѕвезди и притоа важи дека било која врска може да се помине само еднаш (може да сметате дека откако врската ќе биде помината истата исчезнува).

Задача 4 (3)

- Дефинирајте ја состојбата за поставениот проблем.
- Дефинирајте ја почетната и целната состојба.
- Дефинирајте ги операторите (акциите) во проблемот во однос на вашата состојба.
- Кој алгоритам за пребарување ќе го искористите за да обезбедите оптималност на добиеното решение?
- Доколку треба да искористите информирано пребарување, дефинирајте можна хевристика за проблемот.