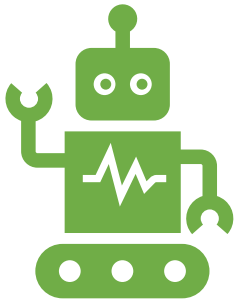
Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Неинформирано пребарување



Аудиториски вежби по курсот
Вештачка интелигенција

2019/2020

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

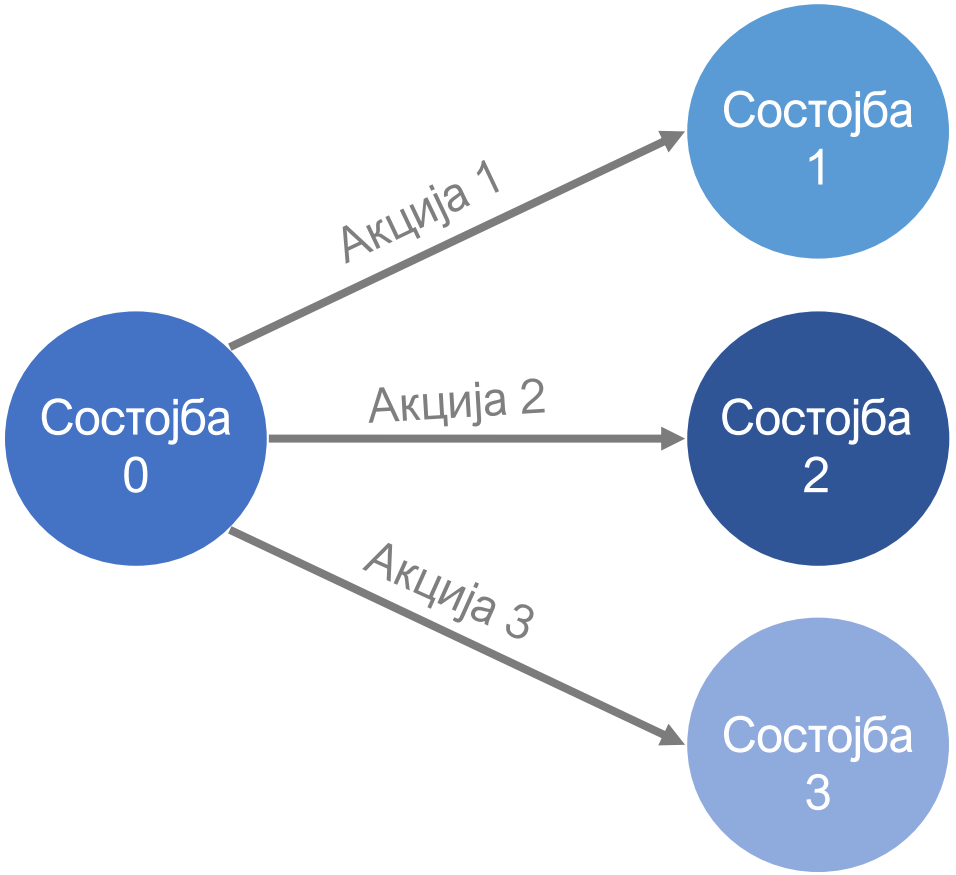
Состојба во Python

- 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

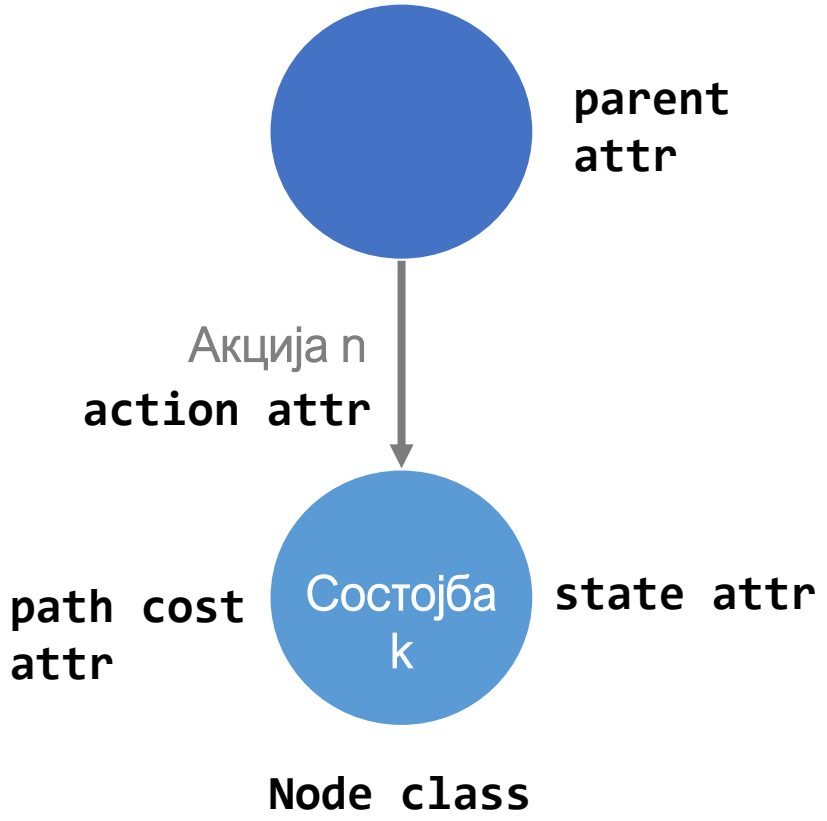


Дефинирање на проблем во Python

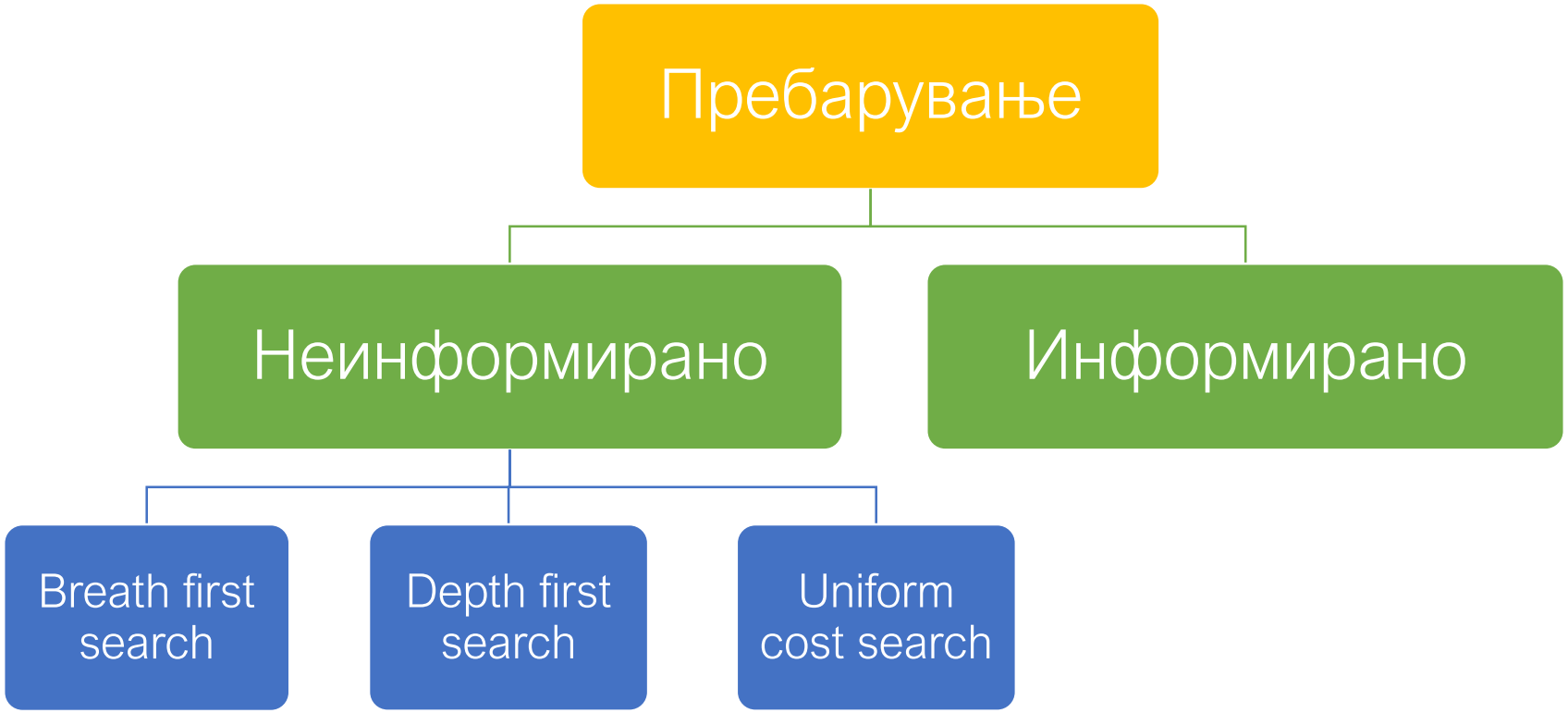
- Дефинираме класа за структурата на проблемот кој ќе го решаваме со пребарување
- Класата Problem е апстрактна класа од која правиме наследување за дефинирање на основните карактеристики на секој еден проблем што сакаме да го решиме

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 неинформирано пребарување



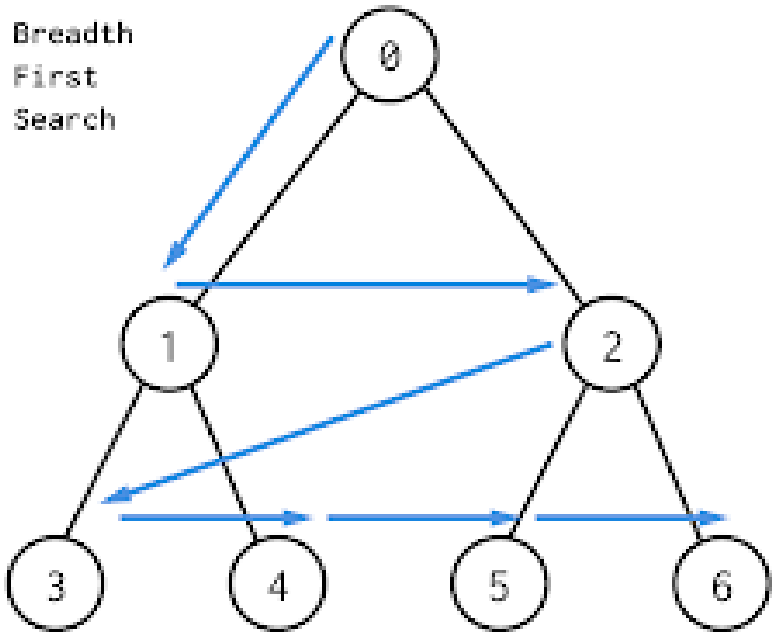
Алгоритми на неинформирано пребарување (2)

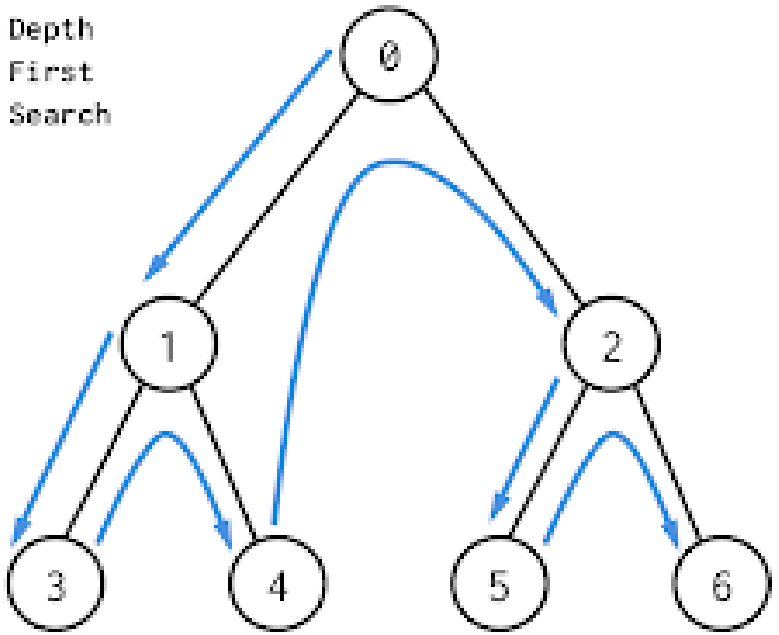
- Секој од алгоритмите за пребарување ќе има:
  - Граф проблем, кој го содржи почетниот јазел S и целниот јазел G.
  - Стратегија, со која се опишува начинот на кој графот ќе се изминува за да се стигне до G.
  - Податочна структура за да се сочуваат сите возможни состојби (јазли) до кои може да се стигне 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.

BFS vs DFS

BFS	DFS
Процесира „ниво по ниво“ посетувајќи ги сите јазли на дадено ниво, по што се продолжува на следното ниво	Следи патека од почетниот јазел до лист јазел, па потоа се следи друга патека од почеток до крај, итн. додека не се посетат сите јазли
Најчесто имплементиран со употреба на редица како податочна структура	Најчесто имплементиран со употреба на стек како податочна структура
Во општ случај побарува повеќе меморија од DFS	Во општ случај побарува помалку меморија од BFS
Оптимален за наоѓање на најкраток пат	Не е оптимален за наоѓање на најкраток пат

BFS vs DFS (2)

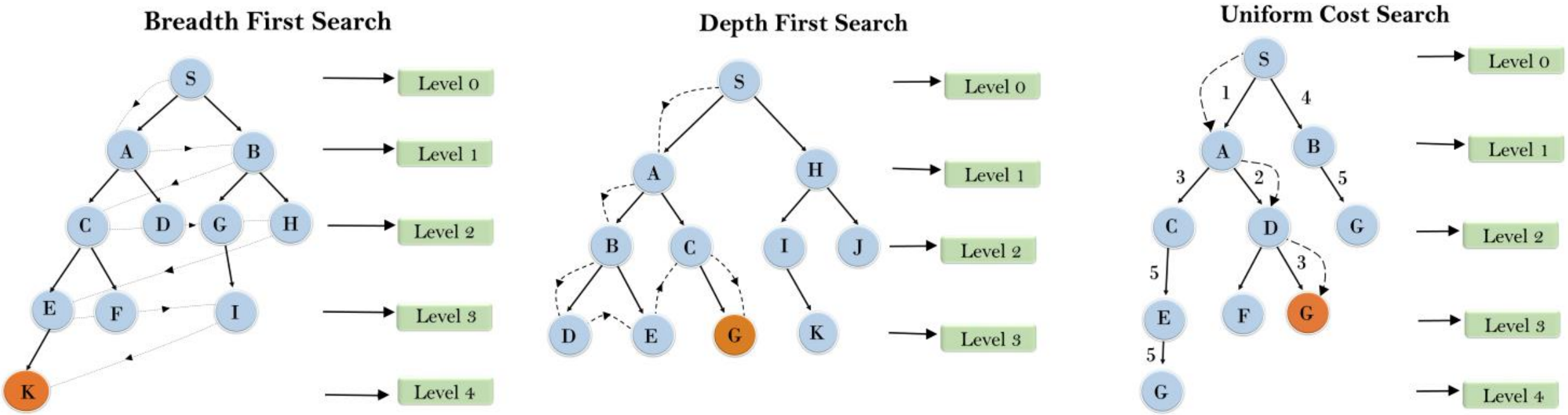




Uniform cost search

- Uniform cost search (UCS) се разликува од BFS и DFS затоа што сега се земаат во предвид цената на чинење за преземање на дадена акција. Односно, истражување на различни гранки може да не ја има истата цена на чинење.
- Целта е да се најде патека, каде што кумулативната сума на цените е најмала.
- Доколку сите цени имаат иста вредност, UCS се сведува на BFS.

Uninformed search example



Проблем: Два сада

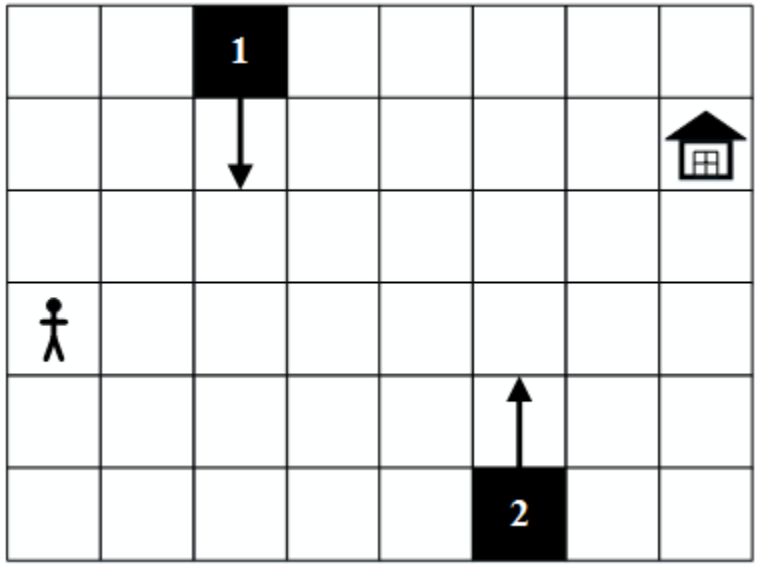
- Дадени се два сада J0 и J1, со капацитети C0 и C1 литри, соодветно. Да се доведат до состојба во која J0 има G0 литри, а J1 има G1 литри.
- Акции:
  - Испразни кој било од садовите
  - Претури течност од еден во друг сад, со тоа што не може да се надмине капацитетот на садот
  - Наполни кој било од садовите (за дома)

Дефиниција на состојба

- Торка (X, Y) која означува дека J0 содржи X литри, а J1 содржи Y литри. Опционална вредност ‘*’, која означува дека е небитно колку литри има во садот.
- Цел: Предефинираната состојба до која сакаме да стигнеме. Ако нè интересира само едниот сад, за другиот можеме да ставиме ‘*’.

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

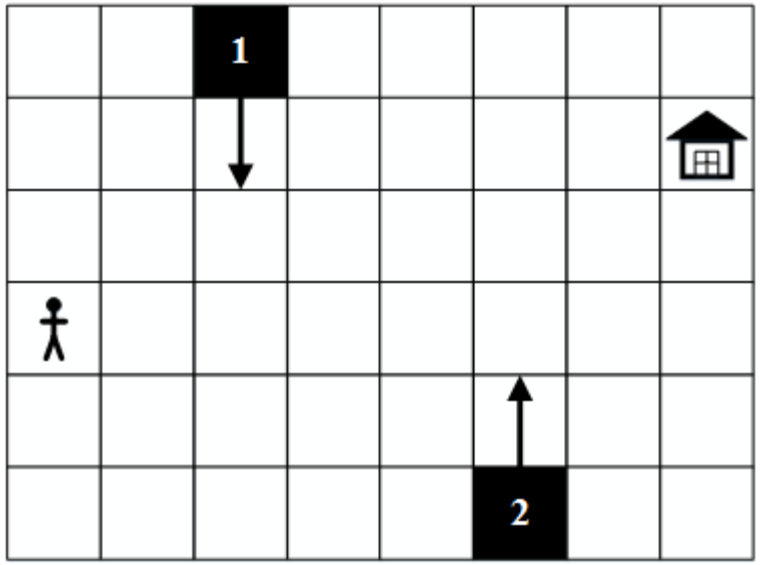


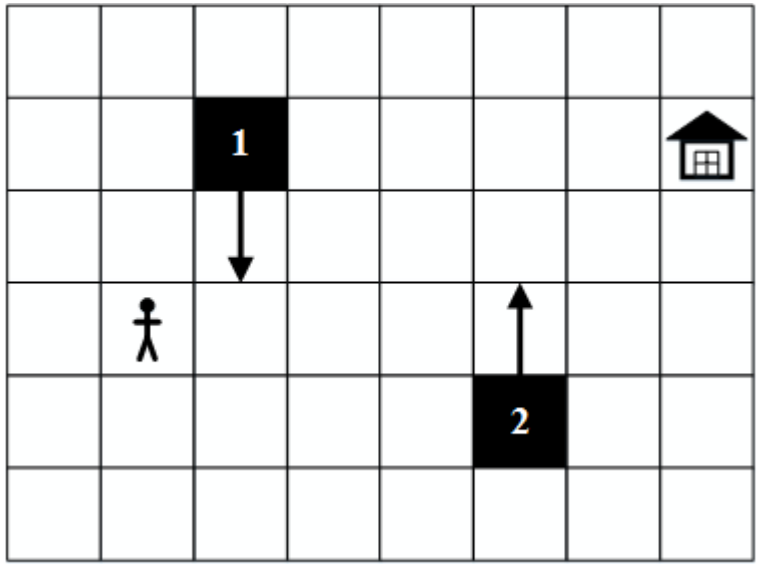
Задача 1 (2)

- 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 неинформирано пребарување. Врз основа на тест примерите треба самите да определите кое пребарување ќе го користите.

Задача 1 – Анализа на проблемот

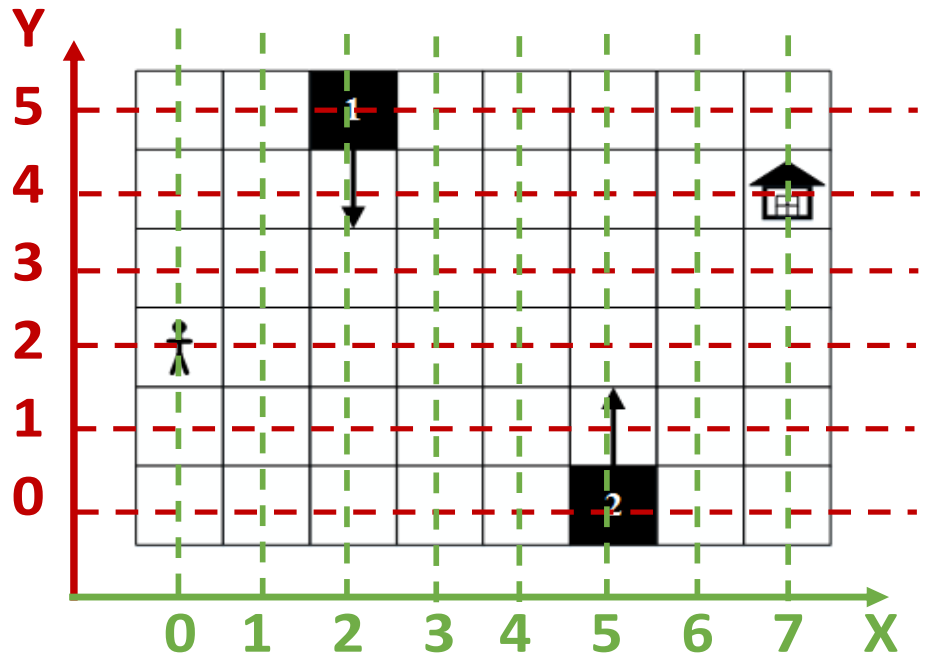
- Првиот чекор во анализата е дефинирањето на состојбата
- Без оглед на проблемот, состојбата мора да содржи променливи преку кои ќе се следат промените кои се случуваат како резултат на акциите (операциите) кои се извршуваат
- Начинот на структурирање на состојбата е личен избор на програмерот
  - Секогаш 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

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

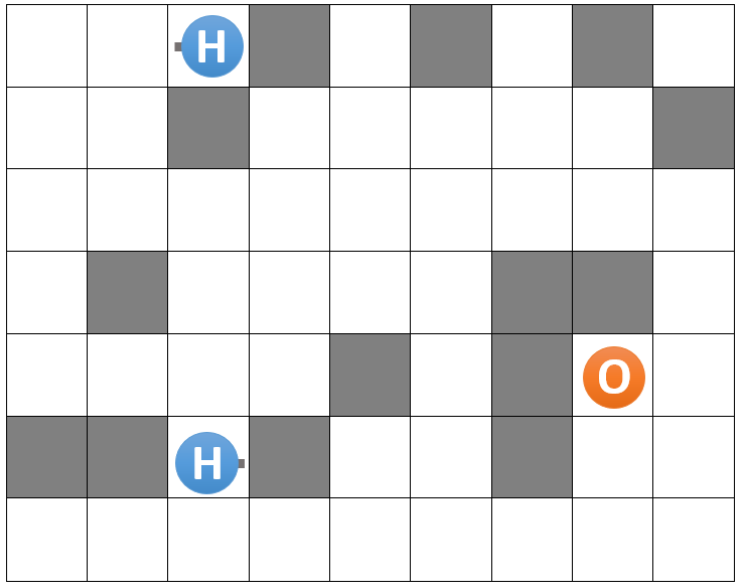
  - Во поедноставениот случај покрај позицијата на човечето треба да ја ажурираме и позицијата на пречките (истото секако треба да се направи и во општиот случај, но надвор од имплементацијата за акциите). Треба да се направат посебни проверки за тоа кога пречките стигаат до крајот на таблата и соодветно 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 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.

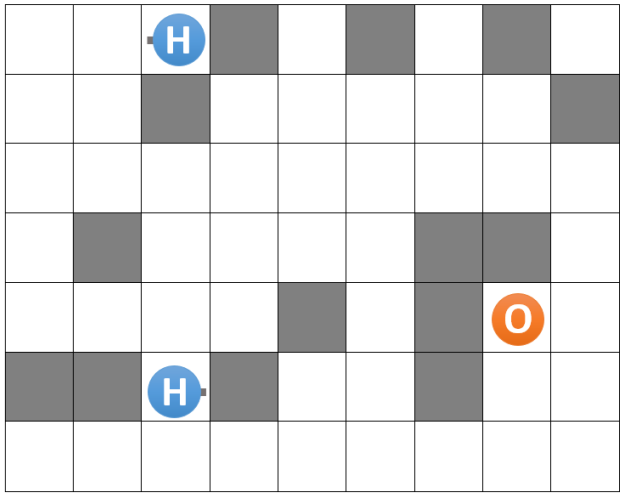


Задача 2 (3)

- 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)

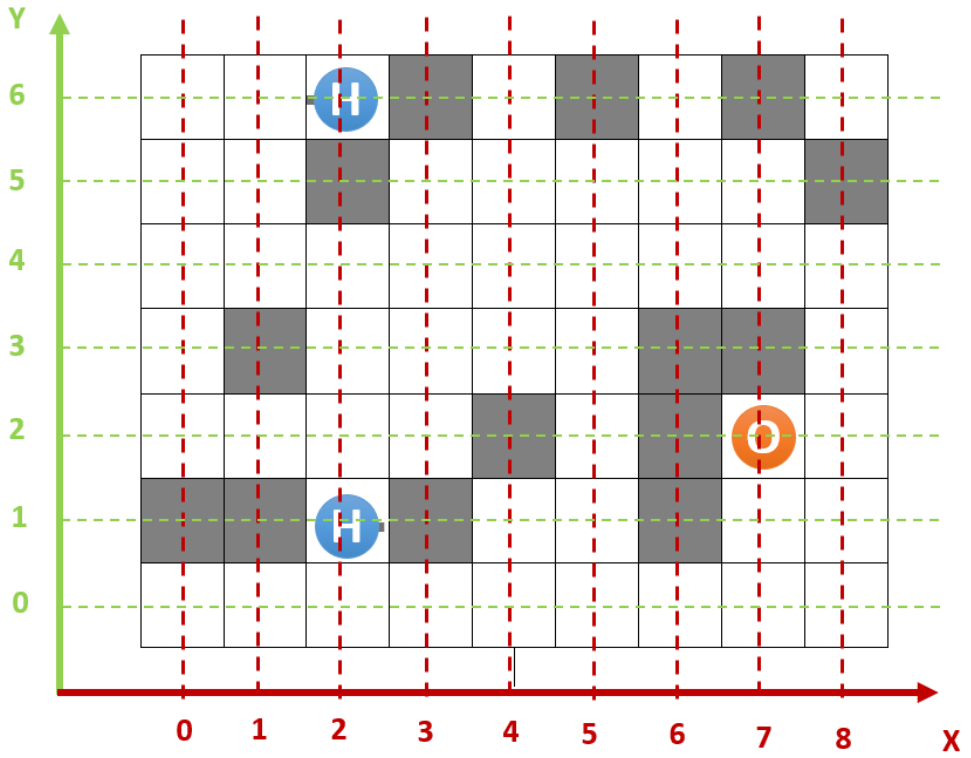
- Движењата на атомите потребно е да ги именувате на следниот начин:
  - RightX - за придвижување на атомот X надесно (X може да биде H1, O или H2)
  - LeftX - за придвижување на атомот X налево (X може да биде H1, O или H2)
  - UpX - за придвижување на атомот X нагоре (X може да биде H1, O или H2)
  - 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 неинформирано пребарување. Врз основа на тест примерите треба самите да определите кое пребарување ќе го користите.

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)

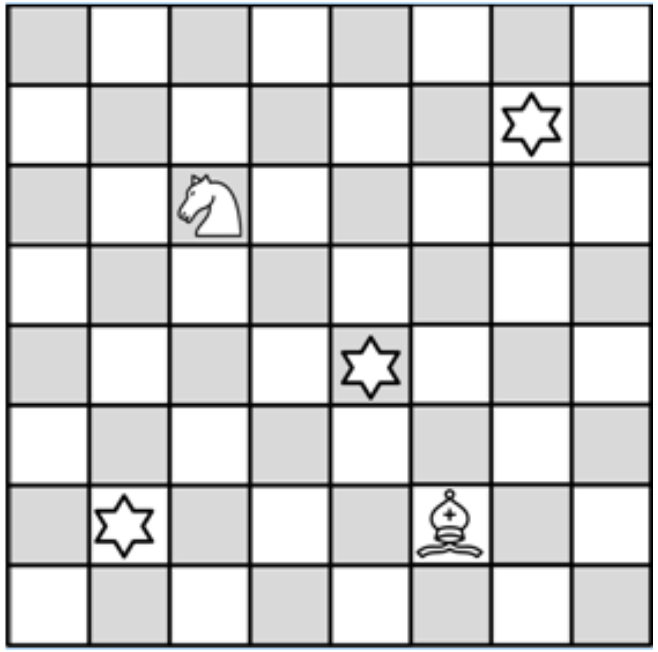
- Почетна состојба
  - За конкретниот пример, почетната состојба ќе биде (2, 1, 7, 2, 2, 6)
- Целна состојба
  - Целната состојба е имплицитна. Не е битна конкретната позиција на молекулите, туку нивната меѓусебна релативна позиција. Треба да се направи функција која ќе проверува дали атомите H1, O,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

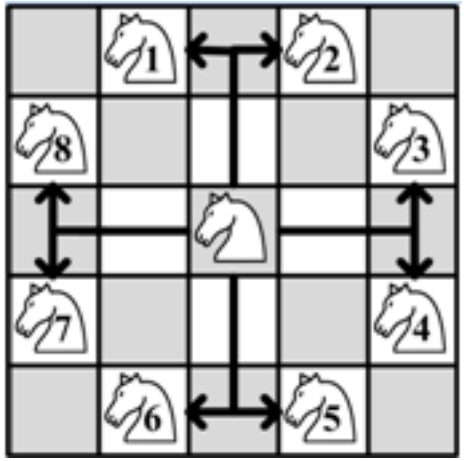
Задача 3: Ѕвезди

- 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.



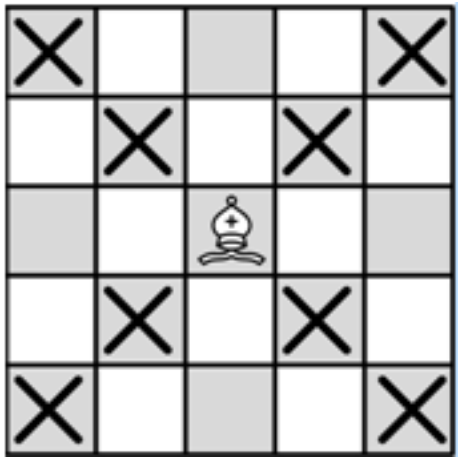
Задача 3 (2)

- На шаховска табла 8x8 поставени се еден коњ, еден ловец и три ѕвезди. Движењето на коњите на шаховската табла е во облик на буквата Г: притоа, од дадена позиција можни се 8 позиции до кои даден коњ може да се придвижи, како што е прикажано на сликата (*1 = горе + горе + лево, 2 = горе + горе + десно, 3 = десно + десно + горе, 4 = десно + десно + долу, 5 = долу + + долу + десно, 6 = долу + долу + лево, 7 = лево + лево + долу, 8 = лево + лево + горе*)



Задача 3 (3)

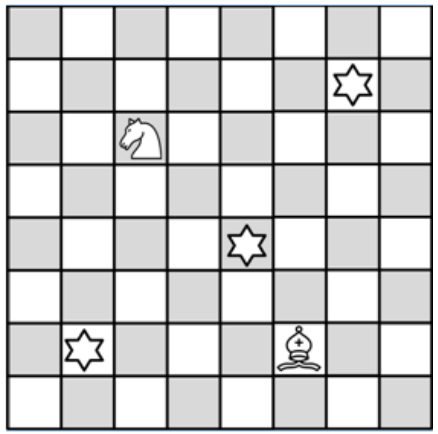
- Движењето на ловците на таблата е по дијагонала. Ловецот прикажан на сликата може да се придвижи на кое било од полињата означени со X.



- Целта на играта е да се соберат сите три ѕвезди. Една ѕвезда се собира доколку некоја од фигурите застане на истото поле каде што се наоѓа и ѕвездата.

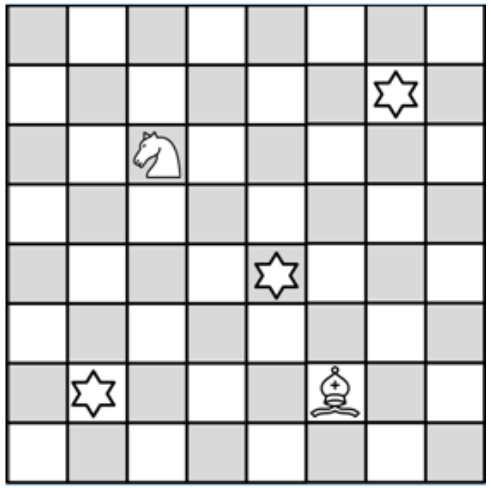
Задача 3 (4)

- Притоа, не е дозволено двете фигури да бидат позиционирани на истото поле и не е дозволено фигурите да излегуваат од таблата. Фигурите меѓусебно не се напаѓаат.
- Движењето на фигурите е произволно, т.е. во кој било момент може да се придвижи која било од двете фигури.
- Потребно е проблемот да се реши во најмал број на потези.



Задача 3 (5)

- За сите тест примери изгледот и големината на таблата се исти како на примерот даден на сликата. За секој тест пример положбите на ѕвездите се различни. Исто така, за секој тест пример се менуваат и почетните позиции на коњот и ловецот, соодветно.
- Во рамки на почетниот код даден за задачата се вчитуваат влезните аргументи за секој тест пример.



Задача 3 (6)

- Движењата на коњот потребно е да ги именувате на следниот начин:
  - K1 - за придвижување од тип 1 (горе + лево)
  - K2 - за придвижување од тип 2 (горе + десно)
  - K3 - за придвижување од тип 3 (десно + горе)
  - K4 - за придвижување од тип 4 (десно + долу)
  - K5 - за придвижување од тип 5 (долу + десно)
  - K6 - за придвижување од тип 6 (долу + лево)
  - K7 - за придвижување од тип 7 (лево + долу)
  - K8 - за придвижување од тип 8 (лево + горе)

Задача 3 (7)

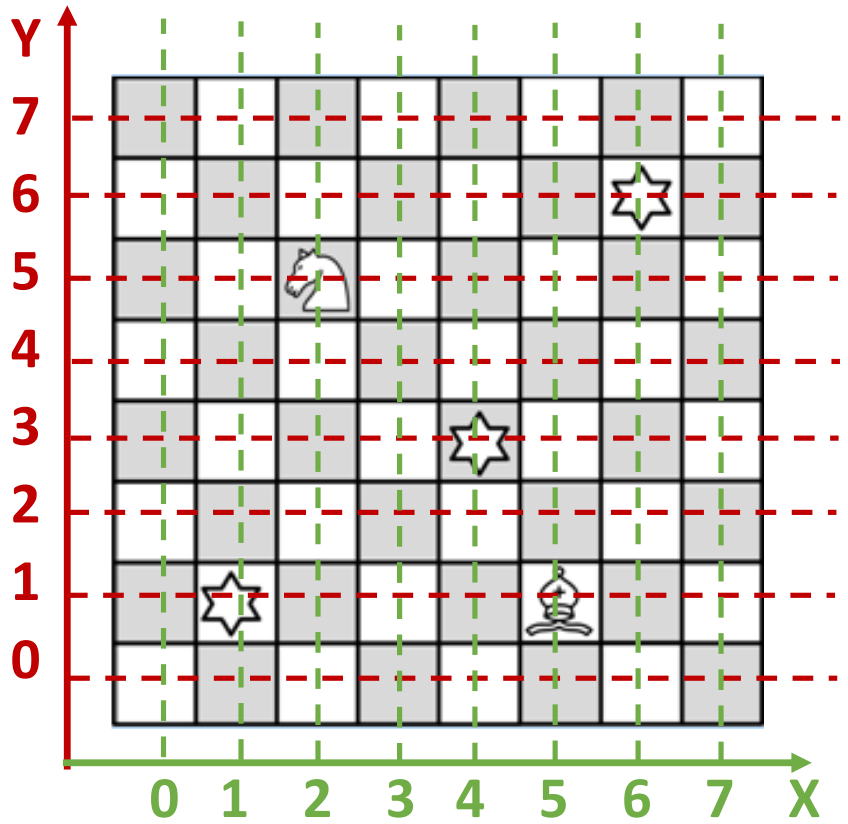
- Движењата на ловецот потребно е да ги именувате на следниот начин:
  - B1 - за придвижување од тип 1 (движење за едно поле во насока горе-лево)
  - B2 - за придвижување од тип 2 (движење за едно поле во насока горе-десно)
  - B3 - за придвижување од тип 3 (движење за едно поле во насока долу-лево)
  - B4 - за придвижување од тип 4 (движење за едно поле во насока долу-десно)

Задача 3 (8)

- 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 фигурите да ги соберат сите три ѕвезди.
- Треба да примените неинформирано пребарување. Врз основа на тест примерите треба самите да определите кое пребарување ќе го користите.

Задача 3 – Анализа на проблемот

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



Задача 3 – Анализа на проблемот (2)

- Во овој проблем ѕвездите, иако се статични, треба да се вклучат во состојбата. Тоа се должи на фактот што ѕвездите се собираат т.е. исчезнуваат и треба да се знае во секој момент дали некаде има ѕвезда или не.
- Промените во проблемот настануваат како резултат на придвижување на шаховските фигури, па нивните позиции ќе бидат дел од состојбата. Состојбата ќе биде торка во која ќе има 4 вредности за координатите на двете фигури и една торка чии елементи се торки од координати на ѕвезди
  - $(X_K,Y_K,X_L,Y_L,L_{STAR})$ каде $LSTAR=((X_{STAR1},Y_{STAR1}),(X_{STAR2},Y_{STAR2}),(X_{STAR3},Y_{STAR3}))$

Задача 3 – Анализа на проблемот (3)

- Почетна состојба
  - За конкретниот проблем, почетната состојба ќе биде
    (2, 5, 5, 1, ((1, 1), (4, 3), (6, 6)))
- Целна состојба
  - Целната состојба е имплицитно зададена. Проблемот е решен кога ќе бидат собрани сите ѕвезди, што значи дека ќе треба да се дефинира функција која ќе проверува дали последниот елемент на состојбата е „празна“ торка

Задача 3 – Анализа на проблемот (4)

- Акции
  - Потребно е да се дефинираат посебни функции за движењето на двете фигури
  - Треба да се внимава на легалноста на следната состојба
    - Да не се излезе од таблата
    - Двете фигури не смеат да бидат на исто поле
  - При генерирањето на следна состојба од тековна треба да се ажурира торката од ѕвезди доколку следната позиција на некоја од фигурите е иста со позиција на постоечка ѕвезда (ѕвездата треба да се избрише)