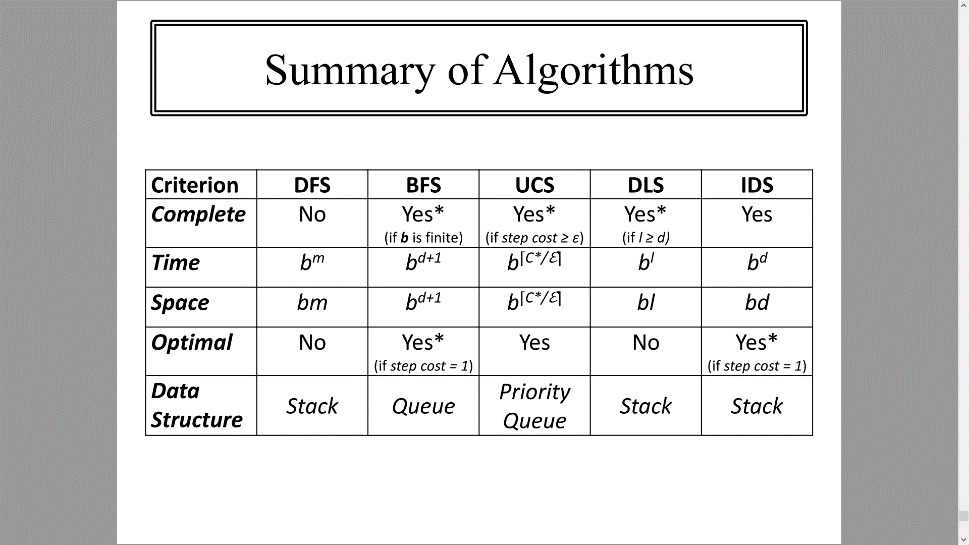
теми за контролно 1

1. **Изкуствен интелект - въведение (тест на Тюринг)**
2. **Интелигентни агенти**
3. **Решаване на проблеми, чрез търсене, Неинформирано търсене**
4. **Информирано търсене**
5. **Локално търсещи алгоритми**
   1. **Hill climbing**
   2. **Simulated annealing**
   3. **Genetic algorithms**
6. **CSP**
7. **Игри**
8. **Knowledge representation**
9. **Expert systems**

* характеристики на алгоритъм
  + пълен (completeness) - дали винаги открива решение, когато такова съществува
  + оптимален (optimality) – пълен алгоритъм,
    - който при повече от едно решение открива най-доброто решението (това с най-малка цена)
    - и ако има едно решение и има повече от един път до него, алгоритъма връща пътя с по-малка цена
  + сложност по време – максималният брой възли, които алгоритъмът обхожда
  + сложност по памет – максималният брой възли, които алгоритъмът пази в паметта в даден момент от време
    - сложностите се мерят в най-лошия случай с голямо О нотация
    - **b -** (branching factor) - максималният брой деца, които може да има дадено състояние
    - **d** – дълбочина на решението с най-малка цена
    - **m** – максимална дълбочина на пространството от състояния
* глобално търсещ алгоритъм
  + ако се налага могат да преминат през всички състояния от пространството от състояния,
* локално търсещи алгоритми
  + те търсят в определено подпространство на пространството от състояния, ако в това подпространство няма решение, то те няма да върнат решение
* неинформирано търсене (DFS, BFS, UCS, DLS, IDS)
  + използва само информация, налична при дефинирането на проблема (в кое състояние се намираме, как може да преминем в следващи състояния и целевото състояния)
  + няма информация дали при преминаване от едно състояние в друго се приближаваме до целта
* информиране търсене (Greedy Best-first search, Beam search, Hill Climbing, A\*)
  + освен информацията от дефиницията на проблема, има оценъчна функция (напр евристика), която дава информация колко дадено състояние е близо до целево състояние, пестят време и памет(понякога)

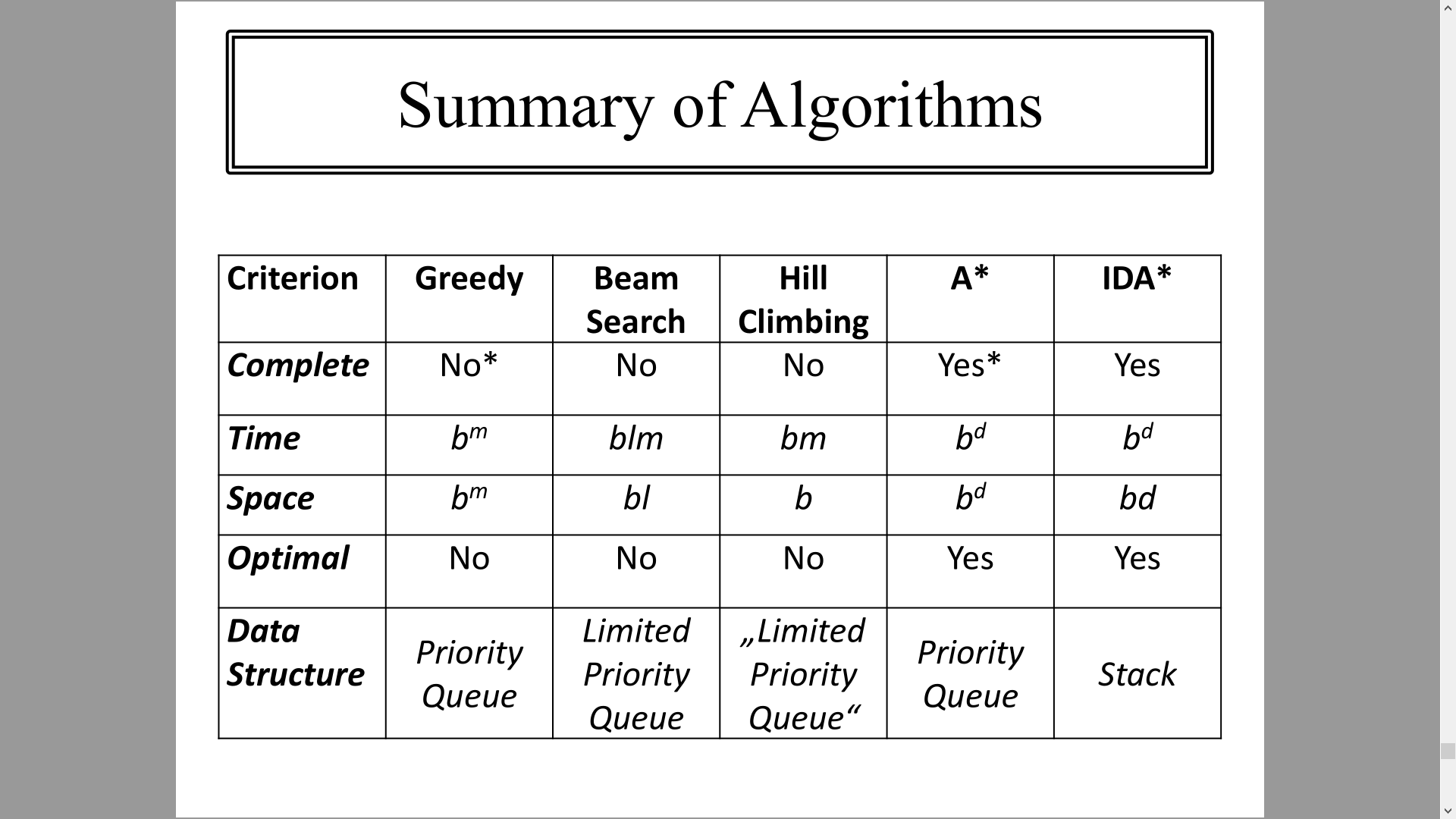
Подробно по теми

1. **Изкуствен интелект - въведение (тест на Тюринг)**
   1. тестът на Тюринг представлява експеримент, при който човек трябва да комуникира с компютърна система. Общуването продължава някакъв период от време и за да бъде тестът успешен, човекът не трябва да може разграничи дали това, с което си е говорил е било машина или човек.
2. **Интелигентни (рационални) агенти**
   1. агент – всяко нещо, което възприема информация от обкръжаващата го среда със сензори (sensors) и въздейства на средата с актуатори (actuators)
   2. интелигентен(рационален) агент – избира действия, чрез които се стреми да максимизира стойността на определена оценка на производителността (performance measure), която се измерва върху средата(environment)
   3. видове агенти
      1. simple reflex (рефлексни агенти)
      2. reflex agents with state (рефлексни агенти с памет)
      3. goal based (-//-, които имат и цел) - могат да прогнозират какво ще стане, при дадено тяхно действие, като това е просто – дали са удовлетворени от даденото действие (напр то води до целта или не)
      4. utility based ( -//- като могат да правят оценка на състояние) – имат utility function, която дава оценка на последиците от действието(по-подробно от goal based агентите)
3. **Решаване на проблеми, чрез търсене, Неинформирано търсене**
   1. типове търсене
      1. offline problem solving – налична е всичката информация за проблема от неговото формулиране (шах)
      2. online problem solving – не е налична всичката информация за проблема от неговото формулиране
   2. типове задачи
      1. deterministic, fully observable - Agent knows exactly which state it will be in; solution is a sequence (шах, табла)
      2. non-deterministic, sensorless - Agent may have no idea where it is; solution (if any) is a sequence
      3. Nondeterministic and/or partially observable – задачи от реалния свят, средата се променя докато решаваме проблема
   3. single state problem – състои се от 4 неща
      1. начално състояние
      2. successor function (оператор, ?функция на прехода)
      3. goal test
      4. path cost – цена за преминаване от едно състояние в друго
   4. state space (пространство от състояния) – мн-вото от състояния, до които може да се достигне (които са достижими) от дадено начално състояние, представя се като граф или дърво
   5. алгоритми за неинформирано търсене

****

* + 1. DFS – използва стек за фронта
    2. BFS – използва опашка за фронта
    3. Uniform Cost Search – използва приоритетна опашка за фронта
    4. Depth Limited Search – DFS с граница на дълбочината
    5. Iterative Deepening Search – пускане на DLS с нарастваща дълбочина, комбинация на DFS и BFS(експоненциална сложност по време, линейна памет, оптималност)

1. **Информирано търсене**



* 1. Best-first search
     1. Greedy Best First Search – фронтът е приоритетна опашка, като оценката на възлите е само стойността на тяхната евристична функция
     2. Beam Search – държи най-добрите l-възела, според тяхната евристика в опашка
     3. Hill Climbing – държи само 1 възел, оценява отново само на база евристиката
  2. A\*
     1. A\* - използва оценъчна функция f = g + h, g e дължината на пътя до възела, h е евристична функция
     2. Memory-bound A\* - А\* където опашката е с лимит
     3. Iterative Deepening Search – използва оценъчната функция f и изпълнява Iterative Deepening Search

1. **Локално търсещи алгоритми**
   1. Hill climbing – преминаване към най-доброто дете от текущото състояние, приключва когато децата на текущото не са по-добри от него.
      1. random restart – против засядане в локален екстремум, пускане на алгоритъма отново със случайно генерирано начално състояние
      2. simulated annealing (stochastic hill climbing) – има параметър „температура“, която в началото на алгоритъма е висока и при преминаване към по-лошо състояние намалява. Оптимизацията спрямо hill climbing, защото позволява и преминаване към по-лоши състояния в началото, с цел да се избегнат локални екстремуми, плата и рамена
   2. Local beam search
      1. вариант на hill climbing, в който има приоритетна опашка с фиксирана дължина, l, в която се държат най-добрите възли. За всеки възел се генерират наследниците и се избират най-добрите k
      2. недостатък – може всичките к възела да заседнат в локален екстремум
   3. Генетични алгоритми
      1. те са stochastic local beam search
         1. stochastic – има случайност при някои от функциите
         2. local – защото разглеждаме само отделни индивиди от популацията
         3. beam search – защото оставяме само най-добрите индивиди и техния брой е фиксиран размер
      2. популация – фиксиран брой случайно генерирани решения, параметрите на решението наричаме гени на индивида
      3. селекция – процес, при който се избират на случаен принцип 2 от най-добрите индивиди от популацията, спрямо тяхната функция на пригодност (fitness function)
      4. кръстосване - ново състояние се генерира от 2 родителски състояния, има няколко варианта – one (two) point, partially mapped
      5. мутация – за всяко дете има малка вероятност за правене на случайна промяна в гените му, например размяна на два гена. Тя е отговорна за поддържане на разнообразието в популацията
2. **CSP**
   1. формулировка
      1. множество на променливите X = {X1, X2, …, Xn}
      2. множество от допустими стойности (фамилия от множества) – D ={D1, D2, …, Dn}, където Di е мн-вото от допустими стойности на променливата Xi, напр Di = {red, green, blue}
      3. множество от ограничения C = {C1, C2, … , Cm}, напр това са X1 + X2 < 3
   2. формулировка 2
      1. дадено е състояния, които са представени от множество от променливи Х със стойности от множество от допустими стойности D
      2. търси си целево състояние, което удовлетворява множеството от ограничения C, които се налагат над подмножества от множеството от променливи и стойностите, които те приемат от домейните им
   3. какви видове ограничения имаме при CSP
      1. унарни - такива, които включват една променлива
      2. бинарни - такива, които включват две променливи
      3. от по-висок ред - такива, които включват повече от две променливи
      4. преференции(soft constraints) – свързани са с цената за присвояване на дадена променлива
   4. какви видове алгоритми се използват за решаване на CSP?
      1. backtracking search (търсене с възврат) – DFS + variable-ordering + fail-on-violation
         1. избиране на променлива - Variable Ordering: Minimum remaining values (MRV) - Choose the variable with the fewest legal left values in its domain
         2. избиране на стойност за присвояване - Value Ordering: Least Constraining Value - Given a choice of variable, choose the least constraining value, I.e., the one that rules out the fewest values in the remaining variables
      2. constraint propagation (разпространение на ограниченията)
         1. arc consistency - подобрение на forward checking, при което се следи дали при присвояване на стойност на една променлива (за удовлетворяване на ограничение) няма да се нарушат други ограничения
      3. local search – min conflicts(hill climbing)

forward checking (проверка напред) – има допълнителна структура, която отразява следи последствията от присвояване на стойност на една променлива върху останалите. Този подход се използва при constraint propagation с подобрения

1. **Игри**
   1. типове игри
      1. игри с перфектна информация –знаем текущото състояние, знаем в какви състояния може да се премине от текущото, знаем състоянията от началото до текущия ход
      2. игри с неперфектна информация – нямаме информация за всичките ходове на противника
      3. детерминистични – имаме информация за всеки възможен следващ ход на играта (вероятностите за всеки ход са равни)
      4. недетерминистични (игри с елемент на шанс) – нямаме пълна информация за следващия ход на противника (зар, размесването на карти)
      5. пълна информация – знаем каква е стратегията на противника и от какво печели той
      6. непълна информация – не знаем какво носи точки на противника
   2. Minimax
      1. идея - избиране на най-добрия резултат срещу оптимален противник
      2. функция на полезността (оценъчна функция) – тя дава оценка на крайните състояния(листата или възлите на определена дълбочина), обикновено >=1 при победа на max, <= -1 при победа на min, 0 при равен
      3. пълнота – алгоритъма е пълен, ако пространството от състояния(в този случай дърво) е крайно
      4. оптималност – да, срещу противник, който играе оптимално
      5. експоненциална сложност по време и линейна памет – като DFS
   3. Minimax with Alpha-beta prunning
      1. числото алфа – най-голямата печалба, който max може да си осигури в текущия път, отначало е -безкрайност
      2. числото бета – най-малката загуба, който min може да си осигури в текущия път, отначало е +безкрайност
      3. алфа-бета отсичането се случва във възел, когато beta <= alpha, т.е. максимизиращият играч е имал по-добра опция по някой предходен клон на дървото
2. **Knowledge representation**
   1. knowledge base
   2. има 4 общи типа за представяне на знания
      1. логики
      2. семантични мрежи
      3. production rules (продукционни правила)
      4. frames
   3. дедукция – от общо към частното
   4. индукция – от частно към общо (от конкретни наблюдения извличаме заключения)
   5. логиги – недостатъци на класическите : не вземат предвид времето (има Temporal logics, които го вземат)
      1. propositional logic (съждителна, пропозиционална логика)
         1. съждения P = it is wet
         2. връзки – и, или, не, следва, еквивалентно
         3. скоби, истина TRUE, лъжа FALSE
         4. по-удобни за разбиране от хората
      2. first order predicate logic (предикатни логики от първи ред)
         1. имат по-голяма изразителна сила от съждителните
         2. имат по-разширен синтаксис от съждителните
         3. по-удобни за разбиране от компютрите
         4. Higher Order Predicate Logic, Fuzzy logic, Multi-valued logics, …
   6. семантични мрежи
      1. представят отношения между обектите чрез различни видове графи
      2. графи –> могат лесно да се представят графично и да са разбираеми от хората
      3. позволява „разсъждения по аналогия“
      4. conceptual graphs (понятийни графи) – вариант на семантичните мрежи
      5. плюсове –
         1. лесно се правят йерархии, лесно се проследяват връзки и са гъвкави
      6. минуси
         1. значенията на възлите може да са неясни (многозначност, може да липсва контекст)
         2. трудно се обозначават и обработват изключения
         3. трудни за програмиране
   7. production rules (продукционни правила)
      1. съдържа двойки <condition, action>
      2. плюсове – прост синтаксис, лесни за разбиране, модулярни(лесно се премахват и добавят правила) и гъвкави
      3. минуси – трудно се моделират йерархии, не са ефективни при големи системи, има знания, които не могат да се изразят с правила
   8. frame representation (фреймово представяне)
      1. информацията се съдържа в рамки със слотове
      2. слотовете могат да връзки към други слотове или действия(програми), които трябва да се изпълнят при определени условия
      3. плюсове - много гъвкави, има огромна изразителна сила, лесна разширяемост, има възможност за специализирани процедури, лесно се справя със липсващи стойности и стойности по подразбиране
      4. минуси – трудни за програмиране
3. **Expert systems**
   1. притежават знания за специфична приложна област
   2. компоненти
      1. user interface (потребителски интерфейс) – за взаимодействие с потребителите
      2. explanation facility (механизъм за обяснения) – защо експертната система дава този отговор. например да покаже списък с факти, които са довели до взетото решение
      3. knowledge base (база от знания) - използва продукционни правила за формализъм за представяне на знания
      4. inference engine (механизъм за разсъждения, извод) – използва информация от работната памет
      5. working memory (работна памет) – всякакви факти, които имат отношение към проблема
      6. agenda (дневен ред, последователност от действия) – приоритизиран списък с правила, които се отнасят за дадената задача
      7. knowledge acquisition facility (механизъм за натрупване на знания) – знанията се взимат от реални експерти за специфична приложна област, може с времето да се коригират и допълват
   3. плюсове
      1. по-голяма наличност
      2. по-ниска цена
      3. multiple expertise (представя знания от много експерти)
   4. с какво се различава от други AI системи
      1. има специфична задача от реалния свят, която има голяма сложност
      2. нужна е висока производителност и бързодействие
      3. трябва да може да обосновава решенията, които взима
4. **? Planning**

неща за контролното

20тина въпроса, 3 типа въпроса

* отворени – напр какво е кръстосване (процес на репродуциране при генетичните алгоритми)
* затворени с 1 верен или повече верни (ясно е посочено)
* задача за обхождане на граф напр

въпроси от асистента на другата група

1. Какво е евристика?

Функция, която оценява колко е добро дадено състояние (колко е близко до финалното). Тази функция трябва да е бърза (по-бърза от колкото точен алгоритъм, който ще намери точно колко далеч сме от крайното състояние). И трябва да ни дава насока кои върхове са по-добри за обхождане.

1. Какво е Best First Search?
   1. Алгоритъм, който приоритизира кандидатите за обработване спрямо някаква метрика (евристика). Примерно А\* е такъв.
2. Какво е Beam Search?
   1. Алгоритъм, който държи само предефиниран константен брой кандидати. Т.е приоритетната опашка е с ограничен размер. Използва се за да пестим памет. Може да правим Beam Search само когато имаме Best First Search.
3. При даден граф да се опише в какъв ред Uniform Search ще посети върховете.
4. Какво е Random Restart?
   1. Рестартиране на алгоритъма с друго произволно начално състояние при даден threshold или след постигане на отговор. Използва се при задачи с локален минимум/максимум, за да избягаме от локалния мин/макс и да търсим глобален такъв, като започнем търсенето на друго произволно място.