Kolokwium nr 2 z przedmiotu: SZTUCZNA INTELIGENCJA **Poznań 14.6.2004 r.**

Imię i nazwisko:		**Nr indeksu**	

W poniższym teście zaznacz wszystkie zdania prawdziwe.

1. Heurystyka h_1 jest lepiej poinformowana niż h_2. Można stąd wyciągnąć wniosek, że

	$\forall n \in N \quad h_1(n) \le h_2(n)$		h_1 i h_2 są monotoniczne
	h_2 odwiedzi mniej wierzchołków niż h_1		h_1 zawiera więcej reguł niż h_2
C	h_1 odwiedzi mniej wierzchołków niż h_2		

2. Przeszukiwanie w tył zaleca się stosować, gdy

	znane są wszystkie dane początkowe	**D**	stan początkowy nie jest dany w sposób jawny
	hipoteza nie jest znana	**E**	wczesna eliminacja celów może wyeliminować przeszukiwanie pewnych gałęzi
C	hipoteza jest dana w sformułowaniu problemu		

3. Przeszukiwanie w przód zaleca się stosować, gdy

	nie wszystkie dane są zawarte w sformułowaniu problemu	**D**	znane są wszystkie dane początkowe
	stan początkowy nie jest dany w sposób jawny	**E**	jest wiele potencjalnych celów, ale tylko kilka możliwości zastosowania faktów wejściowych do konkretnego problemu
C	trudno sformułować hipotezę docelową		

4. Funkcja oceny heurystycznej

	określa odległość stanu bieżącego od celu		jest ciągła
B	z algorytmem best-first-search nazywa się algorytmem A		jest określona na zbiorze łuków A przestrzeni stanów
C	jest określona na zbiorze stanów N		

5. Algorytmem dopuszczalnym jest

A	dowolna heurystyka monotoniczna	**D**	algorytm A*
B	algorytm przeszukiwania wszerz		algorytm A
	algorytm przeszukiwania w głąb		

6. Które z poniższych metod poprawy algorytmu *alfa-beta* są uniwersalne i ich skuteczne zastosowanie nie zależy od rodzaju gry:

	poszukiwanie stanów stabilnych (ang. *quiescence search*)		przeszukiwanie w głąb (ang. *depth first search*)
B	pogłębianie pojedynczych ruchów (ang. *singular extensions*)	**E**	metody z minimalnym zakresem α-β (ang. *null window search*)
	metoda „pustego ruchu" (ang. *null move search*)		

7. Klasyczny algorytm *alfa-beta* przeszukuje przestrzeń stanów gry:

A	w przód		wszerz
	w tył		dowolnie
C	w głąb		

8. Algorytm *alfa-beta* powstał w rezultacie zastosowania w algorytmie *min-max* metody:

	dziel i rządź (ang. *divide and conquer*)		najszybszego spadku
B	podziału i ograniczeń (ang. *branch and bound*)		podziału i cięć (ang. *branch and cut*)
	połowienia przedziału		

9. Który z poniższych algorytmów wymaga zastosowania tablicy przejść (ang. *transposition table*) ze względu na skuteczność i/lub efektywność jego działania:

A	algorytmy z rodziny MTD		metoda „pustego ruchu" (ang. *null move search*)
B	algorytm NegaScout (PVS)		metoda dziel i rządź (ang. *divide and conquer*)
	algorytm Aspiration Search		

10. Wersja *fail-soft* algorytmu *alfa-beta* różni się od wersji klasycznej tym, że zwraca wartość oceny stanu:

	tylko, gdy jest większa lub równa α i mniejsza lub równa β		tylko, gdy jest większa od β i mniejsza od α
B	niezależnie od aktualnego zakresu α-β		równą α, gdy faktyczna wartość jest mniejsza od α lub równą β, gdy faktyczna wartość jest większa od β
	tylko, gdy jest większa od α i mniejsza od β		

11. W wersji *negamax* algorytmu *alfa-beta* funkcja heurystyczna wykorzystywana do oceny stanów:

A	zależy od głębokości na jakiej znajduje się oceniany stan		jest zmienną losową o rozkładzie dwumianowym
	jest taka sama jak w klasycznym algorytmie *alfa-beta*		jest odwrotnością funkcji oceny wykorzystywanej w klasycznym algorytmie *alfa-beta*
	zależy od tego, który gracz wykonuje ruch w korzeniu grafu gry		

12. Które z poniższych algorytmów wykorzystują przeszukiwanie z minimalnym zakresem α-β (ang. *null window search*)?

	Aspiration Search		metoda BFS
	metoda ETC		metoda pogłębiania pojedynczych ruchów (ang. *singular extensions*)
C	metoda „pustego ruchu" (ang. *null move search*)		

13. Algorytm NegaScout opiera się między innymi na wykorzystaniu:

	przeszukiwania z dowolnym zakresem α-β	**D**	ruchów „zabójców" (ang. *killer heuristics*)
B	tablicy przejść (ang. *transposition table*)		przeszukiwania z iteracyjnym pogłębianiem
C	tablicy historii ruchów (ang. *history heuristics*)		

14. Skuteczność zastosowania tablicy przejść (ang. *transposition table*):

A	zależy od głębokości przeszukiwania	**D**	zależy od stanu początkowego (korzenia grafu gry)
	nie zależy od rodzaju gry		zależy od strategii przeszukiwania (w przód/ w tył)
	nie zależy od rozmiaru zajmowanej pamięci		

15. Tablica historii ruchów (ang. *history heuristics*):

A	opiera się na bezkontekstowej ocenie pojedynczego ruchu		jest wykorzystywana do wyboru ruchu tylko w korzeniu grafu gry
	ma wyższe wymagania zasobowe niż tablica przejść	**E**	uzależnia ocenę ruchu wykorzystywaną w ich porządkowaniu od głębokości przeszukiwania
	pełni rolę wyroczni		

16. System reprezentacji wiedzy powinien spełniać następujące wymagania:

A	zdolność reprezentacji		efektywność pozyskiwania wiedzy
	efektywność reprezentacji	**E**	zdolność wnioskowania
C	zdolność pozyskiwania wiedzy		

17. Definicja ekstencjonalna zbioru obiektów:

A	wymaga wylistowania wszystkich elementów zbioru		ułatwia reprezentację wartości domyślnych
	pozwala definiować zbiory nieprzeliczalne		polega na podaniu własności charakteryzujących wszystkie i tylko te obiekty, które należą do zbioru
C	jest praktyczna w przypadku małych zbiorów		

18. Elementy scenariusza to:

A	sceny	**D**	ścieżki
B	role	**E**	warunki wejściowe
C	rekwizyty		

19. Odwzorowanie na reprezentację:

	jest funkcją różnowartościową		jest funkcją monotoniczną
	jest funkcją wzajemnie jednoznaczną		jest funkcją ciągłą
	jest funkcją rzeczywistą		

20. Sieci semantyczne:

A	należą do słabych strukturalnych form reprezentacji wiedzy	**D**	stanowią strukturę typu *slot-and-filler*
B	umożliwiają reprezentację wiedzy deklaratywnej	**E**	zostały zaproponowane do przedstawienia znaczenia słów w języku angielskim
	należą do mocnych strukturalnych form reprezentacji wiedzy		

21. Do strukturalnych form reprezentacji wiedzy zaliczamy:

A	sieci semantyczne	**B**	scenariusze	**C**	ramy
D	zależności pojęciowe	**E**	stereotypy		

22. Generator problemów w ogólnym modelu uczącego się agenta:

A	służy do eksploracji nowych akcji		generuje wzorce działania
	realizuje cele uczenia podane przez element uczący		przekazuje proponowane akcje bezpośrednio do efektorów
C	przekazuje proponowane akcje do elementu wykonawczego		

23. Uczenie indukcyjne

	stosuje zasadę indukcji matematycznej	**D**	jest ogólnym modelem uczenia maszynowego
	opiera się na modelu indukcji magnetycznej	**E**	na podstawie zbioru przykładów funkcji *f*, znajduje funkcję *h* aproksymującą *f*
	opiera się na koncepcji łańcuchów Markowa		

24. Algorytm uczenia drzew decyzyjnych

A	opiera się na zasadzie brzytwy Okhama		bazuje tylko na przykładach pozytywnych
	pozwala znaleźć minimalne drzewo decyzyjne		bazuje tylko na przykładach negatywnych
C	jest algorytmem przybliżonym (heurystyką)		

25. Model konekcjonistyczny

A	składa się z dużej liczby identycznych elementów	**D**	pozwala na równoległe i rozproszone sterowanie siecią
B	reprezentuje wiedzę w postaci wag na połączeniach	**E**	pozyskuje wiedzę w procesie uczenia sieci
C	jest alternatywą dla modeli kognitywistycznych		

26. Pojedynczy liniowy perceptron:

	jest nowszą odmianą sieci Hopfielda	**D**	jego proces uczenia jest zbieżny
	może się nauczyć rozwiązywać dowolny problem klasyfikacji	**E**	może się nauczyć rozwiązywać dowolny liniowo separowalny problem klasyfikacji
	może rozwiązać problem XOR		

27. Twierdzenie Rosenblatta

	zahamowało rozwój sieci neuronowych w latach 50-tych		mówi, że uczenie dowolnej sieci jest zbieżne
	mówi, że pojedynczy perceptron rozwiązuje problem XOR	**E**	mówi, że uczenie jednowarstwowego perceptronu jest zbieżne
	zostało obalone w latach 80-tych		

28. Perceptron wielowarstwowy:

A	może się nauczyć rozwiązywać dowolny problem klasyfikacji	**D**	może rozwiązać problem XOR
	jego proces uczenia jest zbieżny	**E**	składa się z warstwy wejściowej, wyjściowej i warstw ukrytych
	im więcej ma warstw tym więcej może się nauczyć		

29. Sieć Hopfielda:

A	zapamiętuje informację jako wzorzec aktywacji	**D**	jest odporna na błędy pojedynczych węzłów
B	ma skończoną liczbę stanów równowagi	**E**	pozwala na rozproszone i asynchroniczne sterowanie wnioskowaniem
C	jest przykładem pamięci adresowalnej przez zawartość		

30. Efekt generalizacji w uczeniu sieci neuronowych

	jest efektem ubocznym uczenia sieci		można wzmocnić przez zwiększenie liczby wejść sieci
	pojawia się, gdy sieć jest przeuczona	**E**	pozwala prowadzić poprawne wnioskowanie na przykładach spoza zbioru uczącego
	można wzmocnić przez zwiększenie liczby wyjść sieci		

31. Algorytm backpropagation

	jest zbieżny		wymaga, aby funkcja aktywacji była monotoniczna
B	służy do uczenia wielowarstwowych sieci neuronowych		służy do prowadzenia wnioskowania w sieciach neuronowych
C	wymaga, aby funkcja aktywacji była różniczkowalna		