Webinarium przeprowadzone w ramach projektu "Mistrzostwa w Algorytmice i Programowaniu - Uczniowie", **finansowanego przez:**

Podział (n-1; 1)

- Bardzo typowy schemat metody Dziel i Rządź, to podzielenie zadania rozmiaru n na dwie nierówne części: rozmiaru n-1 i 1.
- Jeśli potrafimy przejść od n-1 do n (dodając jedną daną) w fazie scalania, to schemat się bardzo upraszcza:
  - Jeśli n=1, to wykonaj działania kończące
  - Jesli n>1, to wykonaj rekurencyjnie zadanie dla n-1 danych, a potem scal wynik z wynikiem dla 1-elementowej danej

Sortowanie

- Sortowanie n elementów:
  - Jeśli jest jedna dana, to nic nie rób
  - Jeśli jest n>1 danych, to posortuj n-1 elementów i scal wynik z ostatnią daną
- Scalanie polega tu na wstawieniu do posortowanej n-1-elementowej tablicy jednego elementu tak, aby nie zaburzyć porządku.

Sortowanie przez proste wstawianie

```
procedure InsertionSort(var T:tab;k:Integer);
var j:Integer;
begin {sortujemy T[1..k]}
 if k>1 then begin
    InsertionSort(T, k-1);
    j:=k-1; v:=T[k];
    while (j>0) and (T[j]>k) do begin
        T[j+1]:=T[j];
        j:=j-1
    end;
    T[j+1]:=v
 end;
end;
```

Sortowanie przez proste wstawianie iteracyjnie

```
procedure InsertionSort(var T:tab;k:Integer);
var j,k:Integer;
begin
 for k:=2 to n do begin
   j:=k-1; v:=T[k];
   while (j>0) and T[j]>k do begin
       T[j+1]:=T[j];
       j:=j-1
   end;
   T[j+1]:=v
 end;
end;
```

Sortowanie przez proste wstawianie iteracyjnie

```
void InsertionSort(int T[], int n)
//Sortowanie T[0..n-1]; proste wstawianie
int j,k,v;
begin
 for (k=1; k<n; k++){
   j=k-1; v=T[k];
   while (j>=0 && T[j]>v){
        T[j+1]=T[j];
        j=j-1;
   }
   T[j+1]=v;
 }
```

Analiza kosztu

- Mamy tu prostą sytuację:
  - T(1)=0
  - T(n)=T(n-1)+n-1
- To równanie rekurencyjne daje nam rozwiązanie T(n)=(n-1)+(n-2)+...+1, czyli T(n)=n(n-1)/2

Sortowanie przez scalanie

- Jeśli jednak dokonamy podziału na „połowy”, to możemy otrzymać znacznie ciekawszy algorytm.
- Wystarczy posortować połówki, a potem je scalić (koszt scalenia liniowy)

Mergesort

```
procedure mergesort(var T:Tab;l,p:Integer);
var s:Integer;
begin
 if l<p then begin
  s:=(l+p) div 2;
  Mergesort(T,l,s);
  Mergesort(T,s+1,p);
  scal(T,l,s,p);{Scalamy T[l..s] z T[s+1..p]}
end;
```

scal

```
procedure scal(var T:Tab;l,s,p:Integer);
var k1,k2,k:Integer; A:tab;
begin
 k:=l; k1:=l; k2:=s+1;
 while (k1<=s) and (k2<=p) do begin
  if T[k1]<T[k2] then begin
    A[k]:=T[k1]; k1:=k1+1 end
  else begin
    A[k]:=T[k2]; k2:=k2+1 end;
  k:=k+1;
 for k1:=k1 to s do begin
   T[k]:=T[k1]; k1:=k1+1  end;
 for k2:=k2 to s do begin
   T[k]:=T[k1]; k1:=k1+1 end;
end;
```

Analiza kosztu Mergesort

- Liczba porównań dla danych rozmiaru n, to T(n):
  - $T(1)=0$
  - $T(n)=2T(n/2)+n$
- Rozwiązanie tego równania, to funkcja $T(n)=n \log n$

Quicksort

- Algorytm Quicksort też jest szczególnym przypadkiem zasady Dziel i Rządź.
- Tu wybieramy jakiś element (np. A[1]) i względem niego dokonujemy podziału na elementy mniejsze lub równe jemu oraz większe (lub równe). Ten podział realizujemy np. algorytmem flagi polskiej.
- Faza rządzenia, to tylko odpowiednie wywołanie rekursji.

NIM

- Gra w NIM jest wariantem gry w zapałki, zprzy założeniu że wygrywa ten, kto wykona ostatni ruch, czyli weźmie ostatnią zapałkę.

Dowodzenie poprawności procedur i funkcji rekurencyjnych

- Dowodząc poprawność procedur i funkcji rekurencyjnych stosujemy bezpośrednio metodę indukcji – najlepiej ogólnie indukcji noetherowskiej, czyli w zbiorze dobrze ufundowanym.

Indukcja noetherowska

- Załóżmy, że zbiór A jest dobrze ufundowany za pomocą relacji r.
- Niech $Next(x)=\{y\in A: (x,y)\in r\}$
- Jeśli dla formuły jednej zmiennej $\varphi(x)$ określonej w zbiorze dobrze ufundowanym (A,r) zachodzi dla każdego elementu $x\in A$ następujący warunek

$$(\forall y\in Next(x): \varphi(y)) \rightarrow \varphi(x) \qquad (*)$$

to $\forall x\in A: \varphi(x)$

Dowód twierdzenia o indukcji noetherowskiej

- Załóżmy, że w zbiorze dobrze ufundowanym (A,r) zachodzi warunek (*), a jednak istnieje $x \in A$ takie, że $\sim\varphi(x)$. Zatem musi istnieć $x_1 \in Next(x)$ takie, że $\sim\varphi(x_1)$. Ale wtedy istnieje też $x_2 \in Next(x_1)$ takie, że $\sim\varphi(x_2)$, ...itd. Konstruujemy nieskończony ciąg $x=x_0, x_1, x_2, \ldots$ taki, że dla każdego $i>0$ $x_i \in Next(x_{i-1})$ oraz $\sim\varphi(x_i)$. Pal licho to, że $\sim\varphi(x_i)$. Istotne jest to, że skonstruowaliśmy nieskończony łańcuch relacji r. Sprzeczność.

Klasyfikacja pozycji

- W grach bezstronnych pozycje dzielą się na
  - Wygrywające
  - Przegrywające
- Umawiamy się, że mówimy o pozycjach PO wykonaniu ruchu. Jeśli więc po wykonaniu ruchu dajemy przeciwnikowi pozycję przegrywającą, to nasz przeciwnik powinien przegrać, jeśli będziemy grali uważnie do końca. Jeśli natomiast przekażemy mu wygrywającą, a on nie popełni błędu, to wygra.

Pozycje przegrywające w NIM

- (0,0,0) z definicji
- (n,n,0): jeśli jest ruch przeciwnika, to będziemy kopiować jego ruchy, jak echo
- (1,2,3): po każdym ruchu przeciwnika będziemy mogli doprowadzić do układu (n,n,0)
- ... i co jeszcze?

Charakterystyka pozycji

- Pozycja jest przegrywająca, jeśli jest pozycją końcową (czyli nie można wykonać już ruchu) lub nie jest końcową, ale każdy ruch prowadzi do pozycji wygrywającej
- Pozycja jest wygrywająca, jeśli istnieje ruch prowadzący do pozycji przegrywającej
- Żadnych innych pozycji nie ma!

Aby dobrze grać należy...

- ... poznać strategię, czyli mieć metodę oceny, czy pozycja jest wygrywająca, czy przegrywająca. Szczególnie ważna jest umiejętność przechodzenia od pozycji wygrywającej do przegrywającej, bo często ruchów wygrywających nie ma wiele, a w wielu grach jest tylko jeden z bardzo wielu!

Jak określić dobrą strategię?

- Oczywiście można rozważyć zbiór wszystkich możliwych przebiegów partii od danej pozycji i na jego podstawie opracować strategię, ale szczególnie cenne są strategie lokalne, czyli takie, które nie odwołują się do dynamiki rozwoju sytuacji, tylko statycznie dokonują oceny na podstawie samego oglądu pozycji

Układy pozycyjne

- Liczby naturalne możemy zapisywać w innych układach, niż dziesiętny. W układzie o bazie b dysponujemy b cyframi od 0 do b-1 i każda liczba naturalna dodatnia ma jednoznaczne przedstawienie
- $(c_{n-1}...c_1c_0)_b = c_{n-1}b^{n-1}+...+c_1b^1+c_0b^0$
- Na przykład w układzie trójkowym liczba $(121)_3=1*9 + 2*3 + 1 = 16$

Układ binarny

- Układ binarny jest układem pozycyjnym dla b=2. Mamy zatem 2 cyfry: 0 oraz 1 i potęgi dwójki, z których składamy wartość liczby, np.
- $(10101)_2$=1*16 + 0*8 + 1*4 + 0*2 + 1*1=21
- Każdą liczbę dodatnią można tak przedstawić dokładnie w jeden sposób

Strategia gry w NIM

- Twierdzenie
  - Pozycja jest przegrywająca w NIM wtedy i tylko wtedy gdy liczby zapałek zapisane w układzie binarnym i podpisane jedna pod drugą mają w każdej kolumnie parzystą liczbę jedynek.
- Przykładowo pozycja (1,2,3), czyli
  - 01
  - 10
  - 11 JEST PRZEGRYWAJĄCA, BO MA PO 2 JEDYNKI W KAŻDEJ KOLUMNIE

Układy przegrywające w NIM

- Układem parzystojedynkowym nazwiemy taki układ zapałek, że gdy w zapisie binarnym podpiszemy jedna pod drugą liczby zapałek w kolejnych kupkach, to w każdej kolumnie będzie parzysta liczba jedynek
- Przykład: [1, 6, 7] jest układem parzystojedynkowym, bo
- $1 = (001)_2$
- $6 = (110)_2$
- $7 = (111)_2$

Dowód strategii NIM (indukcyjny)

- Pozycja końcowa (0,0,0) spełnia tezę
- Załóżmy, że teza jest spełniona dla wszystkich k<n, gdzie n jest sumą liczb zapałek w kupkach
- Weźmy teraz pozycję mającą n zapałek i rozważmy 2 przypadki:

Sytuacja parzystojedynkowa

- Jeśli mamy pozycję parzystojedynkową, to wykonanie każdego ruchu doprowadzi nas do zaburzenia parzystości w którejś z kolumn oraz do łącznie mniejszej liczby zapałek. Na mocy założenia indukcyjnego taka pozycja jest wygrywająca, a skoro KAŻDY ruch prowadzi do pozycji wygrywającej, to z definicji nasza wyjściowa jest przegrywająca!

Sytuacja nieparzystojedynkowa

- Jeśli nasza pozycja jest nieparzystojedynkowa, to wystarcza znaleźć pierwszą kolumnę od lewej z nieparzystą liczbą jedynek i liczbę, która ją zawiera (może być ich 1 lub 3) uzupełnić “na siłę” do układu parzystojedynkowego. Na pewno otrzymamy mniejszą liczbę, bo najstarszy ruszony bit zmienił się z jedynki na zero. Otrzymana pozycja jest przegrywająca (indukcja), więc nasza jest wygrywająca (definicja).

Czy liczba kupek jest istotna?

- NIE!
- Nigdzie w dowodzie nie zakładaliśmy że zapałki są w 3 kupkach. Ten dowód jest dobry dla dowolnej liczby kupek (w szczególności równej 1)

Sumy gier

- Możemy umówić się, że jeśli mamy kilka gier, to możemy grać w nie jednocześnie z tym samym partnerem na kilku stołach. Ruch polega wtedy na tym, żeby
  - Wybrać stół
  - Na wybranym stole wykonać ruch w grze na nim się znajdującej
- Suma gier się kończy, gdy któryś z zawodników nie może wykonać ruchu

NIM jako suma gier

- Gdy mamy jedną kupkę, wówczas taki NIM nazwiemy jednowymiarowym. Gdy są 3 kupki – trójwymiarowym.
- W szczególności NIM jest sumą trzech jednowymiarowych NIMów.
- Zauważmy, że o ile strategia dla jednowymiarowego NIMa jest prosta, to dla sumy już nie jest taka oczywista.
- Sumy gier zatem w sposób istotny komplikują sytuację.

Funkcja mex

- Dla właściwych podzbiorów liczb naturalnych $A \subset N$ określamy funkcję
  - $mex(A)=min(n \in N: n \notin A)$
- mex od **m**inimal **ex**cluded value
- Przykłady:
  - mex({0,1,2})= 3
  - mex({1,3,4,5,8}) = 0
  - mex(parzyste) = 1
  - itd...

Funkcja Grundy'ego-Sprague'a

- Niech P będzie zbiorem wszystkich możliwych pozycji w grze *G*. Umówmy się, że dla $p \in P$ zbiór Next(p), to zbiór pozycji do których można dojść z p wykonując pojedynczy ruch.
- Na zbiorze wszystkich pozycji gry bezstronnej określamy funkcję
  - G(p)=0 gdy pozycja jest końcowa lub
  - G(p)=mex({G(q): $q \in$ Next(p)}) jeśli końcowa nie jest

Podstawowe twierdzenia

- Jeśli G(p)=0 to albo p jest pozycją końcową, albo dla każdej pozycji $q \in Next(p)$ mamy $G(q) \neq 0$
- Jeśli $G(p) \neq 0$, to istnieje pozycja $q \in Next(p)$ taka, że G(q)=0.
  - ... czyli mówiąc inaczej: pozycja jest przegrywająca wtedy i tylko wtedy, gdy jej funkcja Grundy'ego jest równa 0.

Przykład

- Gra w 15: zaczynamy od 15 zapałek i na przemian zdejmujemy 1,2 lub 3 zapałki. Wygrywa ten, kto weźmie ostatnią.
- Funkcja Grundy'ego dla gry w 15

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
0															

Przykład

- Gra w 15: zaczynamy od 15 zapałek i na przemian zdejmujemy 1,2 lub 3 zapałki. Wygrywa ten, kto weźmie ostatnią.
- Funkcja Grundy'ego dla gry w 15

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
0	1														

Przykład

- Gra w 15: zaczynamy od 15 zapałek i na przemian zdejmujemy 1,2 lub 3 zapałki. Wygrywa ten, kto weźmie ostatnią.
- Funkcja Grundy'ego dla gry w 15

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
0	1	2													

Przykład

- Gra w 15: zaczynamy od 15 zapałek i na przemian zdejmujemy 1,2 lub 3 zapałki. Wygrywa ten, kto weźmie ostatnią.
- Funkcja Grundy'ego dla gry w 15

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
0	1	2	3												

Przykład

- Gra w 15: zaczynamy od 15 zapałek i na przemian zdejmujemy 1,2 lub 3 zapałki. Wygrywa ten, kto weźmie ostatnią.
- Funkcja Grundy'ego dla gry w 15

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
0	1	2	3	0											

Przykład

- Gra w 15: zaczynamy od 15 zapałek i na przemian zdejmujemy 1,2 lub 3 zapałki. Wygrywa ten, kto weźmie ostatnią.
- Funkcja Grundy'ego dla gry w 15

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
0	1	2	3	0	1										

Przykład

- Gra w 15: zaczynamy od 15 zapałek i na przemian zdejmujemy 1,2 lub 3 zapałki. Wygrywa ten, kto weźmie ostatnią.
- Funkcja Grundy'ego dla gry w 15

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
0	1	2	3	0	1	2									

Przykład

- Gra w 15: zaczynamy od 15 zapałek i na przemian zdejmujemy 1,2 lub 3 zapałki. Wygrywa ten, kto weźmie ostatnią.
- Funkcja Grundy'ego dla gry w 15

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
0	1	2	3	0	1	2	3								

Przykład

- Gra w 15: zaczynamy od 15 zapałek i na przemian zdejmujemy 1,2 lub 3 zapałki. Wygrywa ten, kto weźmie ostatnią.
- Funkcja Grundy'ego dla gry w 15

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
0	1	2	3	0	1	2	3	0							

Przykład

- Gra w 15: zaczynamy od 15 zapałek i na przemian zdejmujemy 1,2 lub 3 zapałki. Wygrywa ten, kto weźmie ostatnią.
- Funkcja Grundy'ego dla gry w 15

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3

Czekolada – ostatni wygrywa

G	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
1	0	1	0	2	1	3	0	4	2	5	1	6	3	7	0	8
2	1	0	1	3	0	2	1	5	3	4	0	7	2	6	1	9
3	0	1	0	2	1	3	0	4	2	5	1	6	3	7	0	8
4	2	3	2	0	3	1	2	6	0	7	3	4	1	5	2	10
5	1	0	1	3	0	2	1	5	3	4	0	7	2	6	1	9
6	3	2	3	1	2	0	3	7	1	6	2	5	0	4	3	11
7	0	1	0	2	1	3	0	4	2	5	1	6	3	7	0	8
8	4	5	4	6	5	7	4	0	6	1	5	2	7	3	4	12
9	2	3	2	0	3	1	2	6	0	7	3	4	1	5	2	10
10	5	4	5	7	4	6	5	1	7	0	4	3	6	2	5	13
11	1	0	1	3	0	2	1	5	3	4	0	7	2	6	1	9
12	6	7	6	4	7	5	6	2	4	3	7	0	5	1	6	14
13	3	2	3	1	2	0	3	7	1	6	2	5	0	4	3	11
14	7	6	7	5	6	4	7	3	5	2	6	1	4	0	7	15
15	0	1	0	2	1	3	0	4	2	5	1	6	3	7	0	8
16	8	9	8	10	9	11	8	12	10	13	9	14	11	15	8	0
17	4	5	4	6	5	7	4	0	6	1	5	2	7	3	4	12

Czekolada: ostatni przegrywa

G	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*
1	1	0	1	2	0	3	1	4	2	5	0	6	3	7	1	8	4
2	0	1	0	3	1	2	0	5	3	4	1	7	2	6	0	9	5
3	1	0	1	2	0	3	1	4	2	5	0	6	3	7	1	8	4
4	2	3	2	1	3	1	2	6	0	7	3	4	1	5	2	10	6
5	0	1	0	3	1	2	0	5	3	4	1	7	2	6	0	9	5
6	3	2	3	1	2	0	3	7	1	6	2	5	0	4	3	11	7
7	1	0	1	2	0	3	1	4	2	5	0	6	3	7	1	8	4
8	4	5	4	6	5	7	4	0	6	1	5	2	7	3	4	12	0
9	2	3	2	0	3	1	2	6	0	7	3	4	1	5	2	10	6
10	5	4	5	7	4	6	5	1	7	0	4	3	6	2	5	13	1
11	0	1	0	3	1	2	0	5	3	4	1	7	2	6	1	9	5
12	6	7	6	4	7	5	6	2	4	3	7	0	5	1	6	14	2
13	3	2	3	1	2	0	3	7	1	6	2	5	0	4	3	11	7
14	7	6	7	5	6	4	7	3	5	2	6	1	4	0	7	15	3
15	1	0	1	2	0	3	1	4	2	5	0	6	3	7	1	8	4
16	8	9	8	10	9	11	8	12	10	13	9	14	11	15	8	0	12
17	4	5	4	6	5	7	4	0	6	1	5	2	7	3	4	12	0

Gra w 15 na dwóch stołach

G	***0***	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***
0	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
1	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2
2	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1
3	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
4	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
5	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2
6	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1
7	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
9	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2
10	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1
11	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
12	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
13	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2
14	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1
15	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0

Gra w 15 na dwóch stołach

G	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
1	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2
2	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1
3	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
4	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
5	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2
6	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1
7	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
8	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
9	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2
10	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1
11	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
12	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
13	1	0	3	2	1	0	3	2	1	0	3	2	1	0	3	2
14	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1

Pewna prawidłowość

- Gdybyśmy spojrzeli się na przecięcie i-tego wiersza i j-tej kolumny i porównali wyniki w stosunku do wartości jednowymiarowej funkcji Grundy'ego, to zauważymy, że...

?	*0*	*1*	*2*
0	0	1	2
1	1	0	3
2	2	3	0

xor	*00*	*01*	*10*
00	00	01	10
01	01	00	11
10	10	11	00

Funkcja xor

- Funkcja xor :{0,1} --> {0,1} daje wartość 1 wtedy gdy argumenty są różne, a 0, gdy są równe, czyli ma taką tabelę:
-

xor	*0*	*1*
0	0	1
1	1	0

Funkcja xor na zbiorze N

- Możemy potraktować każdą liczbę jako binarną i xorować kolejne bity argumentów, a następnie odczytać wynik, np.
  - Xor(1,2)=3, bo 10 ¤ 01=11
  - Xor(5,9)=12 , bo 0101 ¤ 1001 = 1100
  - Xor(1000,1001)=1
  - ... itd

Własności funkcji xor

- (x ¤ y) ¤ z = x ¤ (y ¤ z) (łączność)
- x ¤ y = y ¤ x (przemienność)
- x ¤ x = 0
- x ¤ y ¤ x = y (przydaje się przy kodowaniu)

Gry bezstronne na kilku stołach

- Twierdzenie Sprague'a-Grundy'ego:
  - Jeżeli gramy na dwóch stołach w grę bezstronną i dla każdej z gier składowych znamy funkcję Grundy'ego, to funkcja Grundy'ego dla sumy dwóch gier p_1 i p_2 wyraża się wzorem $G(p_1,p_2) = G(p_1) \text{ ¤ } G(p_2)$
  - Prostym wnioskiem, wynikającym z łączności funkcji xor jest uogólnienie tego wyniku dla n stołów: $G(p_1,...,p_n) = G(p_1) \text{ ¤...¤ } G(p_n)$

Strategia dla NIM jako szczególny przypadek tw. o sumie gier

- Widzimy teraz, że funkcja Grundy'ego dla jednowymiarowej gry NIM jest bardzo prosta: G(n)=n. Strategia dla n stołów wylatuje nam, jak z rękawa.
- W rzeczywistości dowód twierdzenia o sumie gier jest bardzo podobny do przytoczonego dowodu o NIM.

I na zakończenie...

- Zadanie: opracować strategię dla antyNIMa, czyli gry w NIM, w której ostatni biorący zapałkę przegrywa.
- Uwaga: wcale nie jest to proste, bo antyNIM nie jest sumą jednowymiarowych antyNIMów.

<Mistrzostwa
w Algorytmice
i Programowaniu>

Projekt „Mistrzostwa w Algorytmice i Programowaniu – Uczniowie” jest finansowany ze środków pochodzących z „Programu Rozwoju Talentów Informatycznych na lata 2019-2029”

Dofinansowanie Projektu: 4.887.850,50 zł

Całkowita wartość Projektu: 5.460.850,50 zł