Teoretyczne podstawy informatyki

Wykład 3b
Kombinatoryka i prawdopodobieństwo

Kombinatoryka i prawdopodobieństwo

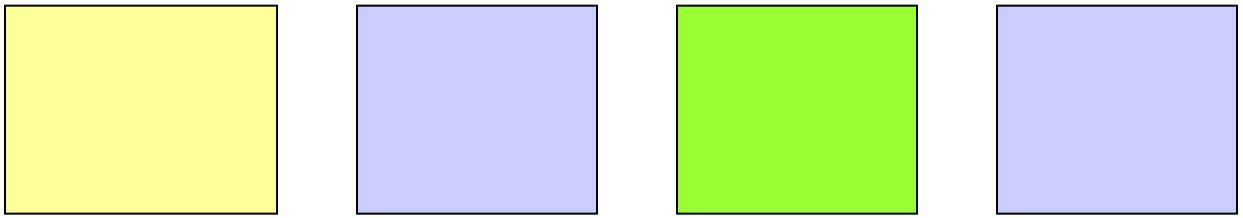
- Często spotykamy się z problemem obliczenia wartości wyrażającej prawdopodobieństwo zajścia określonych zdarzeń.
- Dziedzina matematyki zajmująca się tą tematyką to **kombinatoryka**.
- Pojęcia związane z próbami szacowania prawdopodobieństwa występowania zdarzeń definiuje **teoria prawdopodobieństwa**

➢ *Zacznijmy od kombinatoryki...*

Wariacje z powtórzeniami

Jednym z najprostszych, ale też najważniejszych problemów jest analiza listy elementów, z których każdemu należy przypisać jedną z wartości należących do stałego zbioru.
Należy określić możliwą liczbę różnych przyporządkowań (*wariacji z powtórzeniami*) wartości do elementów.

Przykład:
4 kwadraty, każdy można pomalować jednym z 3 kolorów.
Ile możliwych pomalowań? $3 \cdot 3 \cdot 3 \cdot 3 = 3^4 = 81$



Wariacje z powtórzeniami

Mamy listę n-elementów. Istnieje zbiór k-wartości z których każda może być przyporządkowana do jakiegoś elementu. Przyporządkowanie jest listą n wartości (n_1, n_2,...n_n). Każda z n_1, n_2,... jest jedną z wartości k.
Istnieje k^n różnych przyporządkowań.

Twierdzenie: S(n): liczba możliwych sposobów przyporządkowania dowolnej z k wartości do każdego z n elementów wynosi k^n.

Podstawa:

Przypadek podstawowy to n=1. Jeżeli mamy 1 element możemy wybrać dla niego dowolną spośród k wartości. Istnieje wiec k różnych przyporządkowań. Ponieważ $k^1=k$, podstawa indukcji jest prawdziwa.

Indukcja:

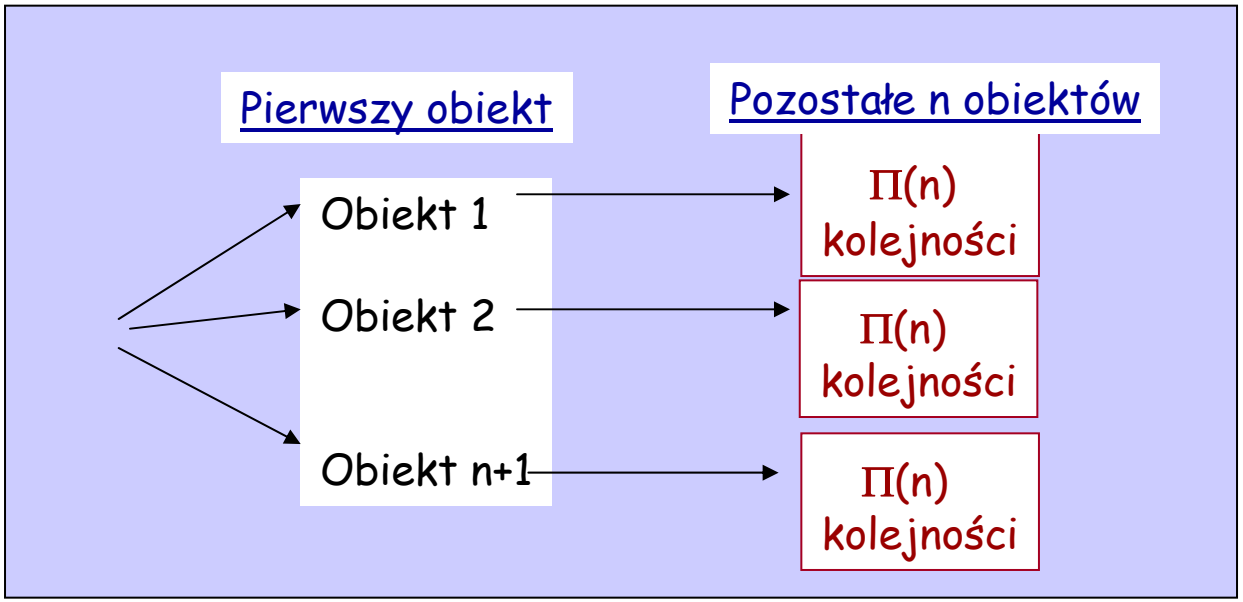
Załóżmy że S(n) jest prawdziwe i rozważmy S(n+1),określające że istnieje k^{n+1} możliwych przyporządkowań jednej z k wartości do każdego z n+1 elementów. Wiemy, że istnieje k możliwości doboru wartości dla pierwszego elementu. Zgodnie z hipoteza indukcyjną, istnieje k^n przyporządkowań wartości do pozostałych n elementów. Łączna liczba przyporządkowań wynosi $k \bullet k^n = k^{n+1}$ cnd.

Permutacje

Mając n różnych obiektów, na ile różnych sposobów można je uporządkować w jednej lini? Takie uporządkowanie nazywamy permutacją. Liczbę permutacji n obiektów zapisujemy jako P(n).

Przykład:

Problem n obiektów (a_1, a_2,....,a_n) które mają zostać posortowane.
Ilość możliwych wyników sortowania jest P(n)



Permutacje n+1 obiektów

Permutacje

Twierdzenie: S(n): P(n) = n! dla wszystkich n ≥ 1

Podstawa:
Dla n=1, S(1) określa że istnieje jedna permutacja dla jednego obiektu.

Indukcja: Załóżmy ze P(n) = n!
Wówczas S(n+1) określają ze P(n+1)=(n+1)!
Rozpoczynamy od stwierdzenia że P(n+1)=(n+1) • P(n)
Zgodnie z hipotezą indukcyjną P(n)=n!, zatem P(n+1)=(n+1)!
Ponieważ n!=n • (n-1) • (n-2) • ... • 1, prawdziwe musi być równanie
(n+1) • n!=(n+1) • n • (n-1) • ... • 1
Iloczyn po prawej stronie jest równy (n+1)! cnd.

- Jednym z interesujących zastosowań wzoru na liczbę permutacji jest dowód na to że **algorytmy sortujące musza działać w czasie co najmniej proporcjonalnym do (n log n), dla n elementów do posortowania**, chyba że wykorzystują jakieś specjalne własności sortowanych elementów.

Wariacje bez powtórzeń

Niekiedy chcemy wybrać tylko niektóre spośród elementów zbioru i nadać im określony porządek.

Uogólniamy opisaną poprzednio funkcję P(n) reprezentującą liczbę permutacji, aby otrzymać **dwuargumentową funkcje P(n,m),** którą definiujemy jako ilość możliwych sposobów wybrania ***m elementów z n-elementowego zbioru***, przy czym ***istotną*** rolę odgrywa ***kolejność wybierania elementów***, natomiast nieważne jest uporządkowanie elementów nie wybranych.
Zatem P(n) = P(n,n).

Przykład:
Ile istnieje sposobów utworzenia sekwencji m liter ze zbioru n liter, jeżeli żadna litera nie może występować więcej niż raz?

Twierdzenie: $S(n)$: $P(n,m) = \frac{n!}{(n-m)!}$ dla wszystkich $m \leq n$

Wyznaczanie liczby kombinacji

Chcemy obliczyć ilość możliwych sposobów wybrania zbioru m elementów z n-elementów, gdy ich uporządkowanie nie ma znaczenia. Taką funkcję zapisujemy zazwyczaj $\binom{n}{m}$ i nazywamy ***n po m*** lub ***kombinacje n elementów z m.***

$$\binom{n}{m} = \frac{P(n,m)}{P(m)} = \frac{n!}{(n-m)! \cdot m!}$$

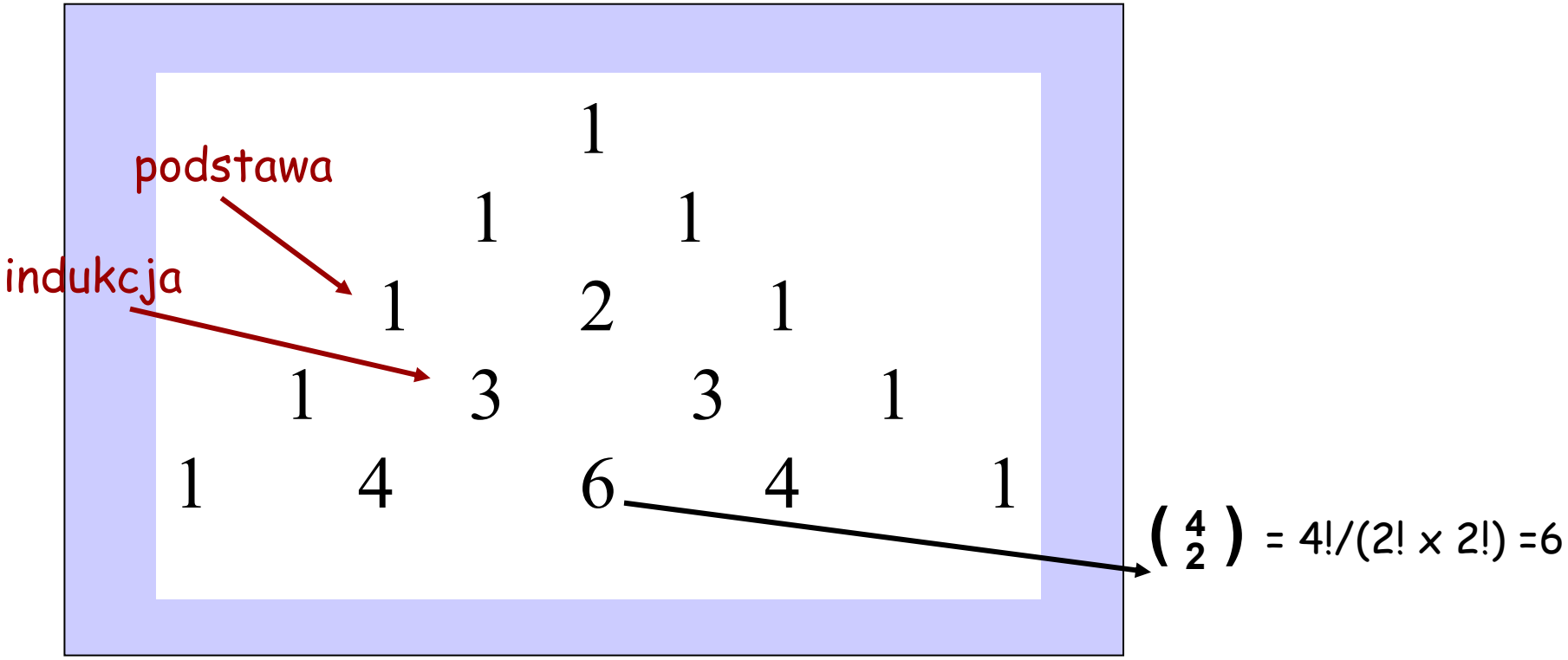
Rekurencyjny algorytm: (ilustruje tzw. trójkąt Pascala)

- **Podstawa:** $\binom{n}{0} = 1$ dla dowolnego $n \geq 1$. Oznacza to że istnieje tylko jeden sposób wybrania zero elementów ze zbioru n-elementowego – wybranie niczego. Także $\binom{n}{n} = 1$, ponieważ jedynym sposobem wybrania n-elementów ze zbioru n-elementowego jest wybranie ich wszystkich.
- **Indukcja:** Jeśli $0 < m < n$, to $\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1}$. Oznacza to, że jeżeli chcemy wybrać m elementów ze zbioru n-elementowego, możemy albo:
  1. nie wybrać pierwszego elementu, po czym wybrać m elementów z pozostałych n-1 elementów. Taką liczbę możliwości wyraża $\binom{n-1}{m}$
  2. wybrać pierwszy element, po czym wybrać m-1 elementów z pozostałych n-1 elementów. Taka liczbę możliwości wyraża $\binom{n-1}{m-1}$.

Trójkąt Pascala

Tą rekurencję często ilustruje się przy pomocy ***trójkąta Pascala***.

$\binom{n}{m}$ = (m+1) liczba w (n+1) wierszu

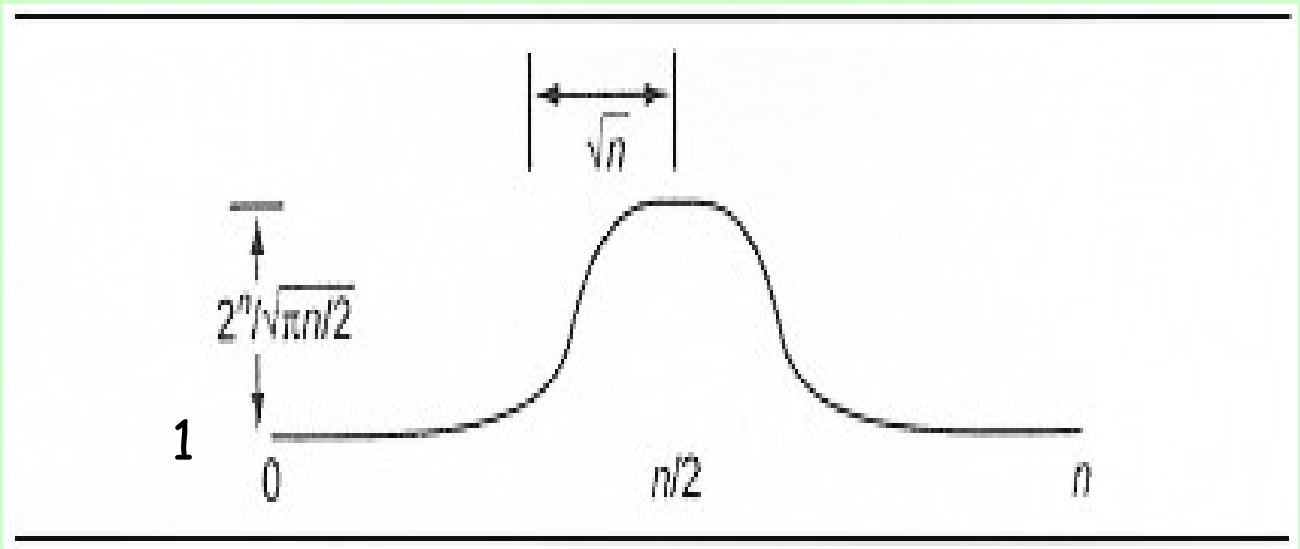


Interesujące własności funkcji $\binom{n}{m}$

⇒ To również są współczynniki rozkładu dwuwyrazowego wielomianu (dwumianu) $(x+y)^n$

⇒ $\Sigma_{m=0}^{n} \binom{n}{m} = 2^n$

⇒ wykres funkcji $\binom{n}{m}$ dla stałej dużej wartości n:



Permutacje z powtórzeniami

Twierdzenie:
S(k): Jeżeli istnieje n elementów podzielonych na k grup o rozmiarach równych odpowiednio i_1, i_2, i_3,...i_k, gdzie elementy jednej grupy są identyczne, ale elementy różnych grup różnią się od siebie, liczba uporządkowań tych elementów wynosi $n! / \Pi_{j=1} i_j!$

Podstawa: Dla k=1, istnieje tylko jedna grupa zawierająca identyczne elementy, które możemy uporządkować tylko w jeden sposób, niezależnie od liczności tego zbioru. Jeśli k=1, to i_1=n, zatem S(1)=n!/n!=1 jest prawdziwe.

Indukcja: Załóżmy że S(k) jest prawdziwe i rozważmy sytuacje, w której mamy k+1 grup. Niech ostatnia grupa składa się z m=i_{k+1} elementów, występujących na m pozycjach, z których możemy je wybierać na $\binom{n}{m}$ sposobów.
Stosując hipotezę indukcyjną otrzymujemy że $S(k+1)=\binom{n}{m} \cdot (n-m)! / \Pi_{j=1} i_j!$
co łatwo można przekształcić (pamiętając że m=i_{k+1}) do postaci:
$S(k+1)= n! / \Pi_{j=1}^{k+1} i_j!$ a więc cnd.

- Jest to typowy problem dla układania anagramów

Łączenie reguł kombinatorycznych

Typowy problem kombinatoryczny wymaga łączenia przedstawionych reguł (cegiełek) w bardziej skomplikowane struktury.

Techniki których używamy to:

- prowadzenie obliczeń jako sekwencji wyborów;
- prowadzenie obliczeń jako różnicy innych obliczeń (np. wszystkich wyborów – nieprawidłowych wyborów);
- prowadzenie obliczeń jako sumy rozwiązań dla podprzypadków które są wzajemnie rozłączne.

Teoria prawdopodobieństwa

Teoria prawdopodobieństwa, szeroko stosowana we współczesnej nauce, ma również wiele zastosowań w informatyce.

- szacowanie czasu działania programów dla przypadków ze średnimi, czyli typowymi danymi wejściowymi
- wykorzystanie do projektowania algorytmów „podejmujący decyzje” w niepewnych sytuacjach, np. najlepsza możliwa diagnoza medyczna na podstawie dostępnej informacji
- algorytmy typu Monte Carlo
- różnego rodzaju symulatory procesów
- ***prawie zawsze „prawdziwe” rozwiązania***

Teoria prawdopodobieństwa

Przestrzeń probabilistyczna:
Skończony zbiór punktów, z których każdy reprezentuje jeden z możliwych wyników doświadczenia. Każdy punkt x jest związany z taką nieujemną liczbą rzeczywistą zwaną prawdopodobieństwem x, że suma prawdopodobieństw wszystkich punktów wynosi 1.
Istnieje także pojęcie nieskończonych przestrzeni probabilistycznych ale nie mają one większego zastosowania w informatyce.

Zdarzenie E:
Podzbiór punktów w przestrzeni probabilistycznej.
Prawdopodobieństwo zdarzenia, P(E), jest sumą prawdopodobieństw punktów należących do tego zdarzenia.

Dopełnienie zdarzenia $\bar{E}$:
Zbiór punktów przestrzeni probabilistycznej które nie należą do zdarzenia $\bar{E}$.
P(E) + P(E) = 1.

Teoria prawdopodobieństwa

Prawdopodobieństwo warunkowe:

E, F są dwoma zdarzeniami z przestrzeni probabilistycznej. Interesuje nas prawdopodobieństwo zajścia zdarzenia F pod warunkiem że zaszło zdarzenie E, **P(F/E).**

Zdarzenia niezależne:

Rzucamy dwoma kostkami, wyrzucenie liczby „1" na pierwszej kostce (zdarzenie E) nie wpływa na możliwość pojawienia się liczby „1" na drugiej kostce (zdarzenie F). **P(F/E)=F(F)**

Zdarzenia zależne:

Ciągniemy dwa razy kartę z talii kart. Wyciągniecie jako pierwszej karty asa (zdarzenie E), wpływa na możliwość wyciągnięcia jako drugiej karty asa (zdarzenie F). **$P(F/E) \neq P(F)$.**

W niektórych sytuacjach liczenie prawdopodobieństw jest łatwiejsze jeżeli podzielimy przestrzeń probabilistyczna na rozdzielne obszary R_i. Wówczas **$P(E) = \sum_{i=0}^{k} P(E/R_i)\, P(R_i)$.**

Przykład z kartami

Ciągniemy dwie karty z talii 52 kart. Liczba punktów w tym doświadczeniu (czyli wariacji bez powtórzeń) wynosi 52 * 51 = 2652.
Zdarzenie E to wyciągniecie jako pierwszej karty As'a.
Liczba punktów to 4 * 51 = 204. P(E) = 204/2652 = 1/13.
Prawdopodobieństwo wyciągnięcia As'a jako drugiej karty, (zdarzenie F), jeżeli pierwsza wyciągnięta karta to był As jest P(F/E) = 3/51 = 1/17. P(E)*P(F/E)=1/13*1/17=1/221.

Podzielmy przestrzeń na dwa obszary:
R1= pierwsza karta to jest as, liczba punktów 4*51=204
R2= pierwsza karta to nie jest As, liczba punktów 2652-204=2448
$P(E \wedge F)=P(E \wedge F/R1)P(R1)+P(E \wedge F/R2)P(R2)$
$P(E \wedge F/R2) = 0$
$P(E \wedge F/R1)=3/51=1/17$; $P(R1)=204/2252=1/13$
$P(E \wedge F)= 1/17*1/13=1/221$

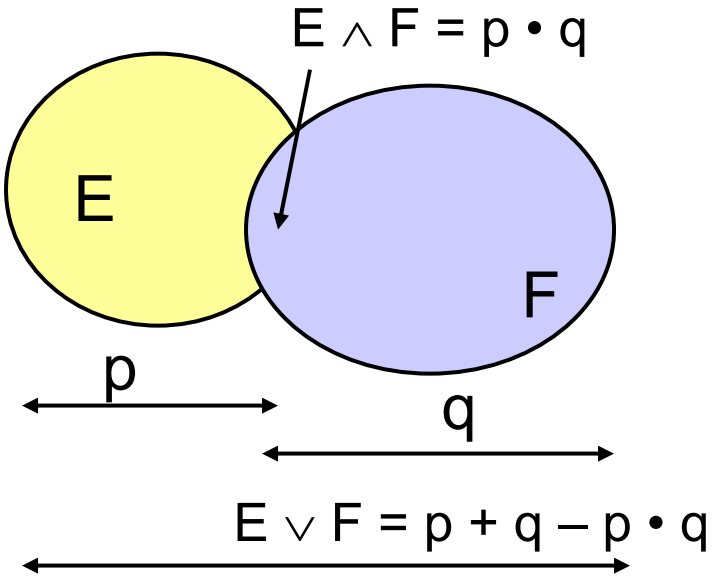
Gdybyśmy po wyciągnięciu pierwszej karty zwracali ja z powrotem do talii, to mielibyśmy również P(F/E) = P(F) = P(E) = 1/13
Wówczas P(E)*P(F/E)=1/13*1/13=1/169

Reguły związane z wieloma zdarzeniami

$P(E) = p,\ P(F) = q$

$\max(0,(p+q-1)) < P(E \wedge F) < \min(p,q)$

$P(E \vee F) = p+q - p \bullet q$



W zastosowaniach czasem akceptujemy że nie możemy wyznaczyć dokładnie prawdopodobieństw oraz zależności między zdarzeniami. Potrafimy tylko wskazać sytuacje najmniej lub najbardziej prawdopodobne.

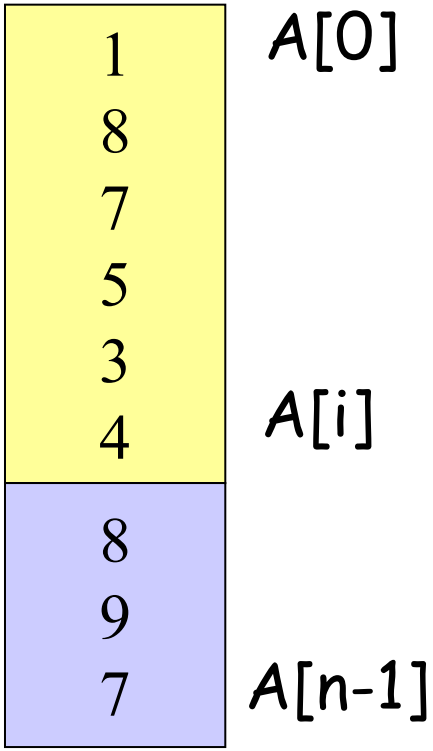
- ***Zastosowanie: różnego typu diagnostyka***

Oczekiwane wartości obliczeń i analiza probabilistyczna

Przypuśćmy, że mamy pewną funkcję określoną na przestrzeni probabilistycznej f(x). Wartość oczekiwana tej funkcji po wszystkich punktach przestrzeni $< f > = \Sigma f(x) P(x)$.

Mamy tablicę n liczb całkowitych, sprawdzamy czy jakaś liczba całkowita „x" jest elementem tej tablicy. Algorytm przegląda całą tablice, po napotkaniu A[i] = x kończy działanie.
Jeżeli A[0] = x to algorytm $O(1)$
Jeżeli A[n-1] = x to algorytm $O(n)$

$< f > = \Sigma (c\, i + d) \bullet 1/n = c\,(n-1) / 2 + d$
$< f > \sim c\, n/2$ dla dużego n



Algorytmy wykorzystujące prawdopodobieństwo

- Jest bardzo wiele różnych typów algorytmów wykorzystujących prawdopodobieństwo.
- Jeden z nich to tzw. **algorytmy Monte-Carlo** które wykorzystują liczby losowe do zwracania albo wyniku pożądanego („prawda”), albo żadnego, („nie wiem”). Wykonując algorytm ***stałą*** liczbę razy, możemy rozwiązać problem, dochodząc do wniosku, że jeśli żadne z tych powtórzeń nie doprowadziło nas do odpowiedzi „prawda”, to odpowiedzią jest „fałsz”. Odpowiednio dobierając liczbę powtórzeń, możemy dostosować prawdopodobieństwo niepoprawnego wniosku „fałsz” do tak niskiego poziomu, jak w danym przypadku uznamy za konieczne.

➢ **NIGDY** jednak nie osiągniemy prawdopodobieństwa popełnienia błędu na poziomie ZERO.

Co to są liczby losowe?

Mówimy, że wyniki pewnych doświadczeń są ***losowe,*** co oznacza ze wszystkie możliwe wyniki są równoprawdopodobne.

Przykładowo, jeżeli rzucamy normalną (prawidłową) kostką do gry to zakładamy że nie ma możliwości fizycznego kontrolowania wyniku tego rzutu w taki sposób aby jeden wynik był bardziej prawdopodobny od drugiego.

Podobnie zakładamy że mając uczciwie potasowana talie kart, nie możemy wpłynąć na wynik - prawdopodobieństwo otrzymania w rozdaniu każdej karty jest identyczne.

Co to są liczby losowe?

Wszystkie generowane przez komputer ***losowe*** sekwencje są wynikiem działania specjalnego rodzaju algorytmu zwanego **generatorem liczb losowych** (ang. Random number generator). Zaprojektowanie takiego algorytmu wymaga specjalistycznej wiedzy matematycznej. Przykład prostego generatora który całkiem dobrze sprawdza się w praktyce to tzw. "**liniowy generator kongurencyjny**".
Wyznaczamy stałe $a \geq 2$, $b \geq 1$, $x_0 \geq 0$ oraz współczynnik $m > \max(a,b,x_0)$.
Możemy teraz wygenerować sekwencje liczb x_0, x_1, x_2, … za pomocą wzoru:

$$x_{n+1} = (a\, x_n + b) \bmod (m)$$

Dla właściwych wartości stałych a,b,m oraz x_0, sekwencja wynikowa będzie wyglądała na losową, mimo ze została ona wygenerowana przy użyciu konkretnego algorytmu i na podstawie "jądra" x_0.

- Dla szeregu zastosowań istotna jest odtwarzalność sekwencji liczb losowych

Algorytmy wykorzystujące prawdopodobieństwo

Mamy pudełko w którym jest n-procesorów, nie mamy pewności czy zostały przetestowane przez producenta. Zakładamy że prawdopodobieństwo że procesor jest wadliwy (w nieprzetestowanym pudelku) jest 0.10.

Co możemy zrobić aby potwierdzić czy pudełko dobre?

- przejrzeć wszystkie procesory -> algorytm O(n)
- losowo wybrać k procesorów do sprawdzenia -> algorytm O(1)

Błąd polegałby na uznaniu ze pudełko dobre (przetestowane) jeżeli nie było takie.

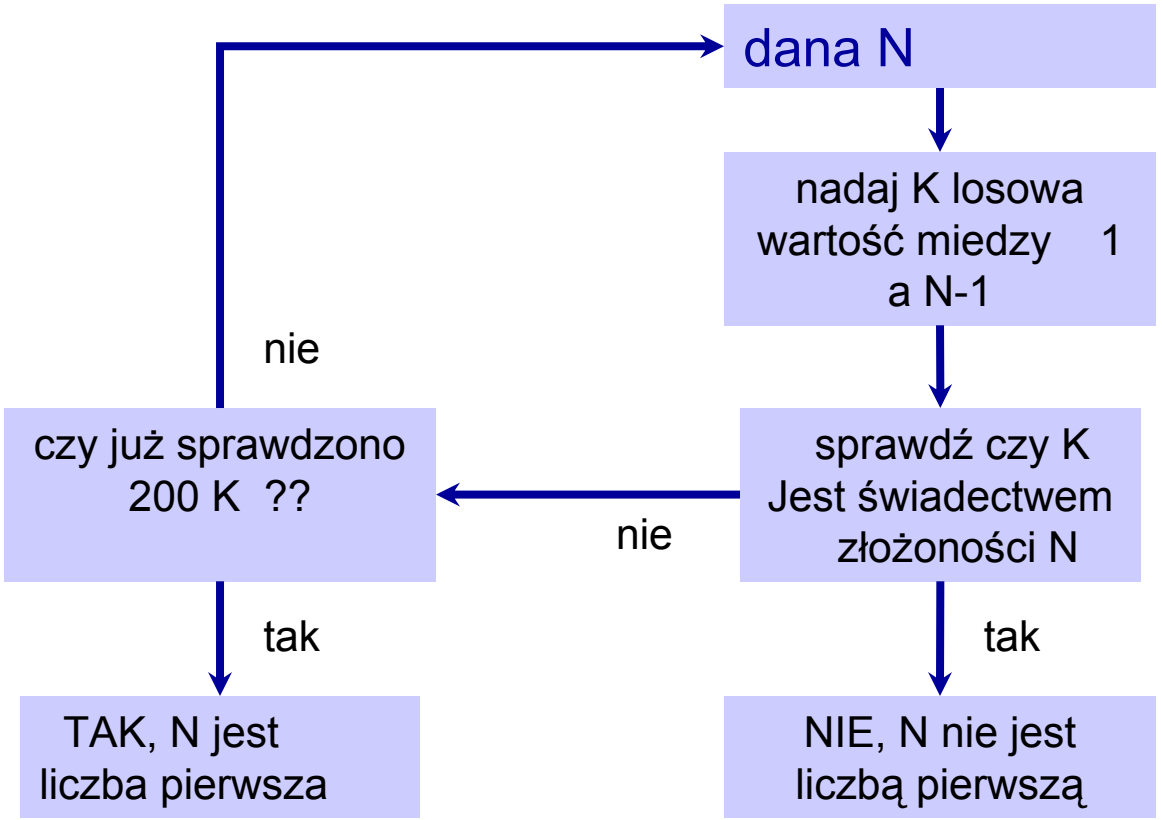
Losujemy k=131 procesorów. Jeżeli procesor dobry odpowiedz „nie wiem”.
Prawdopodobieństwo ze „nie wiem” dla każdego z k-procesorów $(0.9)^k = (0.9)^{131} = 10^{-6}$.
To jest prawdopodobieństwo że pudełko uznamy za dobre choć nie było testowane przez producenta. Za cenę błędu = 10^{-6}, zamieniliśmy algorytm z O(n) na O(1).
Możemy regulować wielkość błędu/czas działania algorytmu zmieniając k

Probabilistyczne algorytmy sprawdzania

„ Czy liczba N jest liczbą pierwszą ?"

- W połowie lat 70-tych odkryto ***dwa bardzo eleganckie probabilistyczne algorytmy*** sprawdzające, czy liczba jest pierwsza. Były one jednymi z pierwszych rozwiązań probabilistycznych dla trudnych problemów algorytmicznych. Wywołały fale badań które doprowadziły do probabilistycznych rozwiązań wielu innych problemów.
- Oba algorytmy wykonują się w czasie wielomianowym (niskiego stopnia), zależnym od liczby cyfr w danej liczbie N (czyli *O* (log N)).
- Oba algorytmy są oparte na losowym szukaniu pewnych rodzajów potwierdzeń lub ***świadectw złożoności*** liczby N.
- Po znalezieniu takiego świadectwa algorytm może się bezpiecznie zatrzymać z odpowiedzią ***„nie, N nie jest liczbą pierwszą",*** ponieważ istnieje bezdyskusyjny dowód że N jest liczbą złożoną.
- Poszukiwanie musi być przeprowadzone w taki sposób aby w pewnym dosadnym czasie algorytm mógł przerwać szukanie odpowiadając, że N jest liczbą pierwszą z bardzo małą szansą omyłki.
- Trzeba zatem znaleźć dająca się ***szybko sprawdzać*** definicje świadectwa złożoności.

„Czy liczba N jest liczbą pierwszą ?”



jeśli to jest odpowiedź to jest to prawda z błędem $1/2^{200}$

jeśli to jest odpowiedź to jest to prawda

Świadectwa złożoności (zarys)

- ⇒ każda liczba parzysta poza 2 to jest złożona
- ⇒ jeżeli suma cyfr jest podzielna przez 3 to jest złożona (iteracyjny prosty algorytm liniowo zależny od liczby cyfr)
- ⇒ ***opiera się o twierdzenie Fermata***
  jeśli N jest liczbą pierwszą oraz K jest dowolna liczba całkowita (1, N-1), to K^{N-1}/N daje resztę równą 1. Co więcej zdarza się (z wyjątkiem kilku złych liczb złożonych), że jeśli K wybierzemy losowo z przedziału (1, N-1), to prawdopodobieństwo tego że K^{N-1}/N da resztę inna niż 1 jest mniejsze niż ½. Liczby złożone spełniają warunek testu dla danej a z prawdopodobieństwem nie mniejszym niż ½.
- ⇒ jeśli K i N nie maja wspólnych dzielników (co by było świadectwem złożoności) policz
  $X=K^{(N-1)/2}$ (mod N), Y=Js(N,K)= +-1 symbol Jacobiego,
  jeśli X = Y to K jest świadectwem złożoności.

Podsumowanie

⇒ Przestrzeń probabilistyczna składa się z punktów z których każdy reprezentuje wynik jakiegoś doświadczenia. Każdy punkt x związany jest z nieujemna liczbą zwaną prawdopodobieństwem punktu x. Suma prawdopodobieństw wszystkich punktów składających się na przestrzeń probabilistyczna wynosi 1;

⇒ Zdarzenie jest podzbiorem punktów z przestrzeni probabilistycznej. Prawdopodobieństwo zdarzenia jest sumą prawdopodobieństw należących do niego punktów. Prawdopodobieństwo każdego zdarzenia mieści się w przedziale od 0 do 1;

Podsumowanie

⇒ Reguła sum określa, że prawdopodobieństwo tego, że zajdzie jedno z dwóch zdarzeń E lub F jest większe lub równe większemu z prawdopodobieństw obu zdarzeń, ale nie większa niż suma tych prawdopodobieństw. Reguła iloczynów określa, ze prawdopodobieństwo tego, że wynikiem pewnego doświadczenia będą dwa zdarzenia E i F, jest nie większe niż mniejsze z prawdopodobieństw obu zdarzeń.

⇒ Wykonując algorytm Monte Carlo stałą liczbę razy, możemy rozwiązać problem, dochodząc do wniosku, że jeśli żadne z tych powtórzeń nie doprowadziło nas do odpowiedzi „prawda”, to odpowiedzią jest „fałsz”.