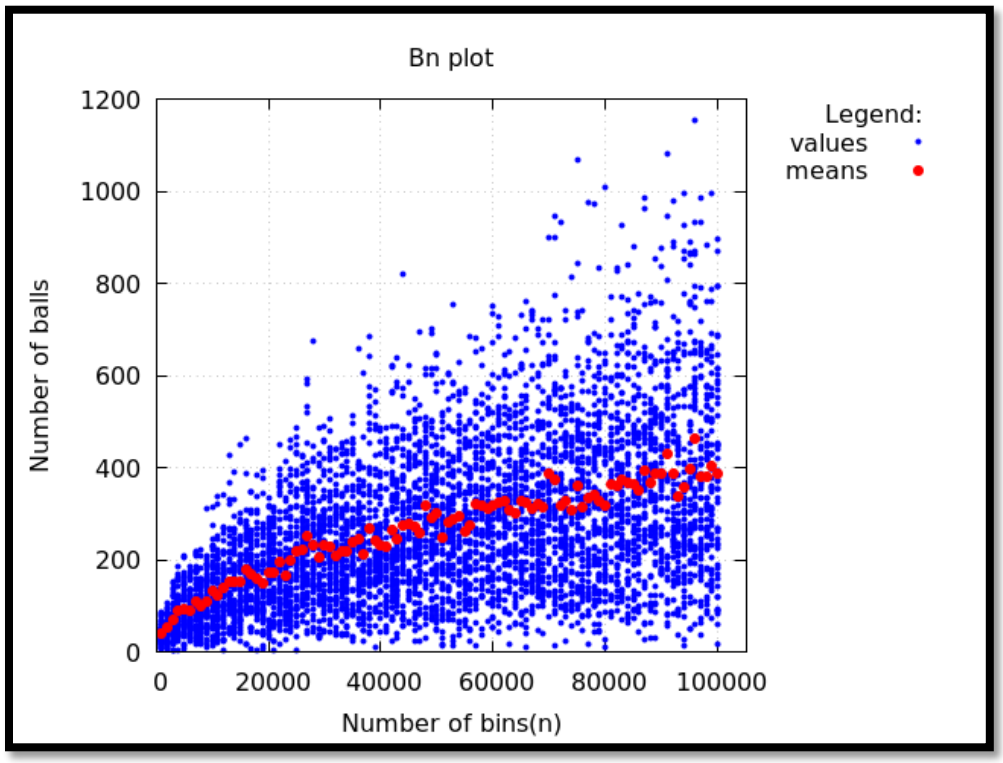
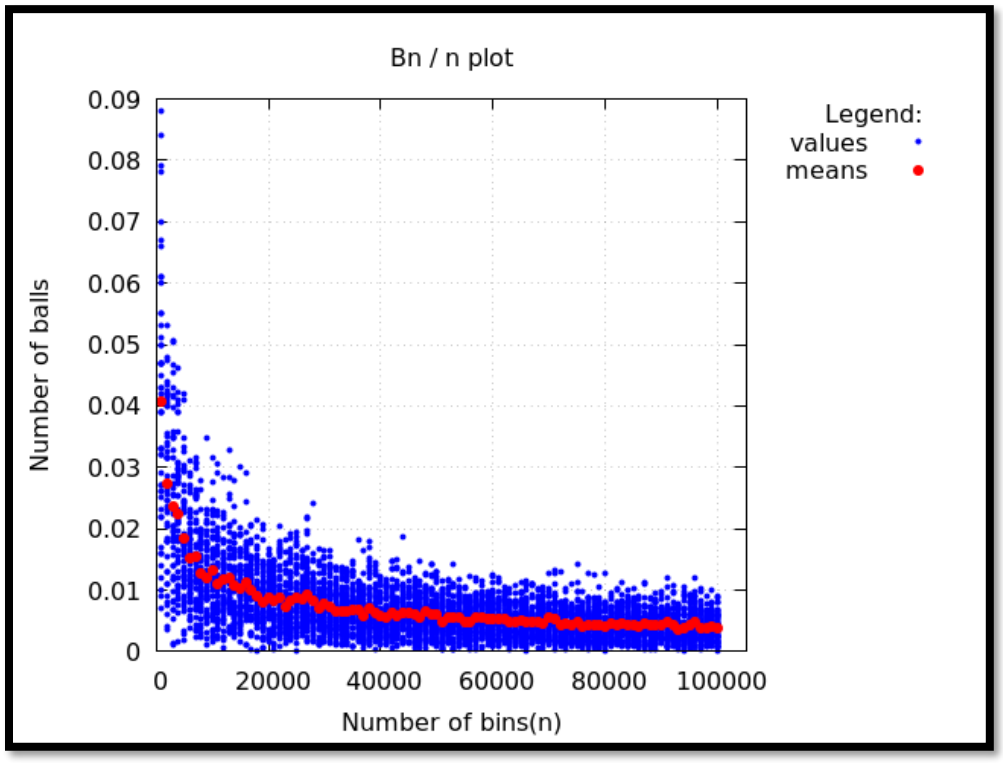
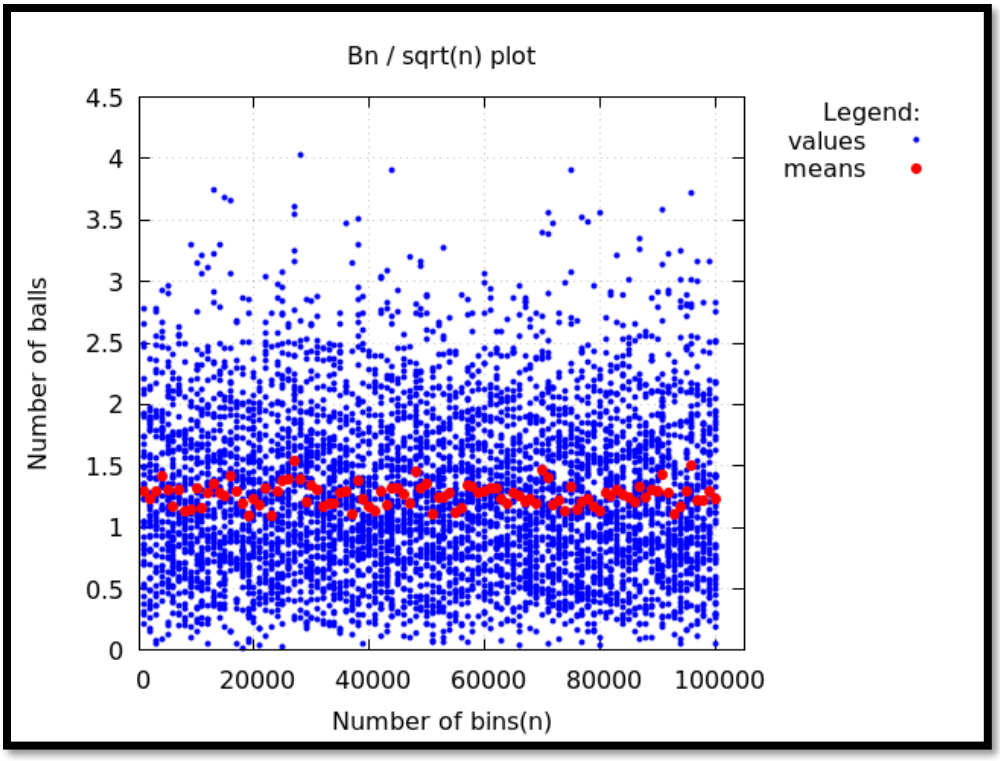
Kule i urny

Bn:





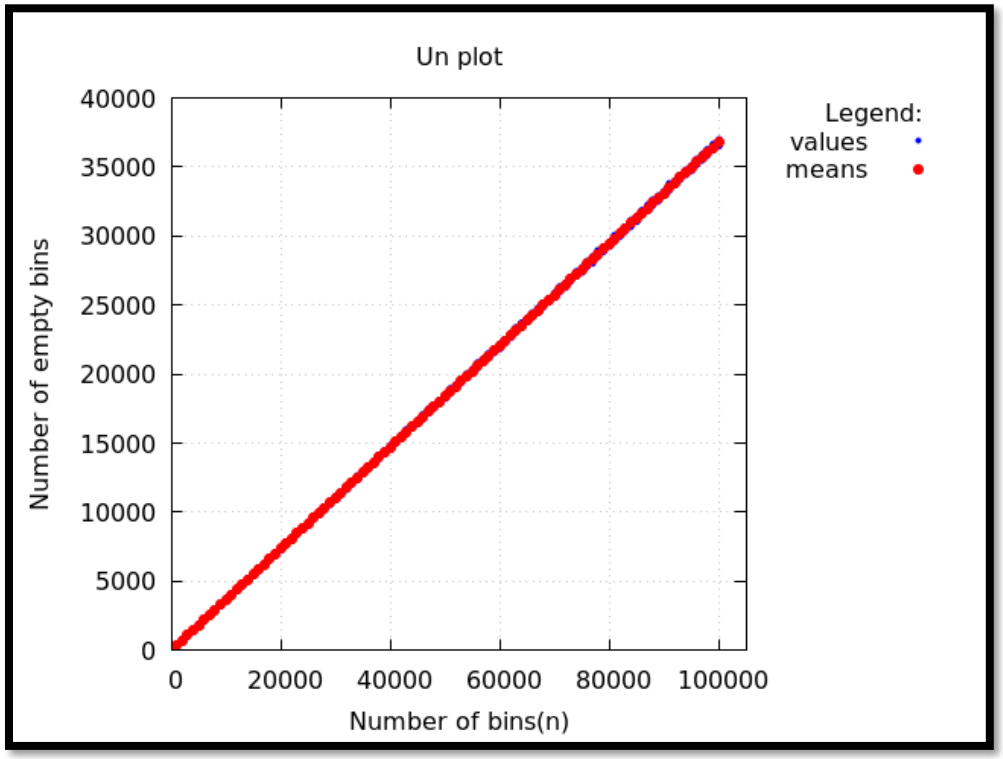


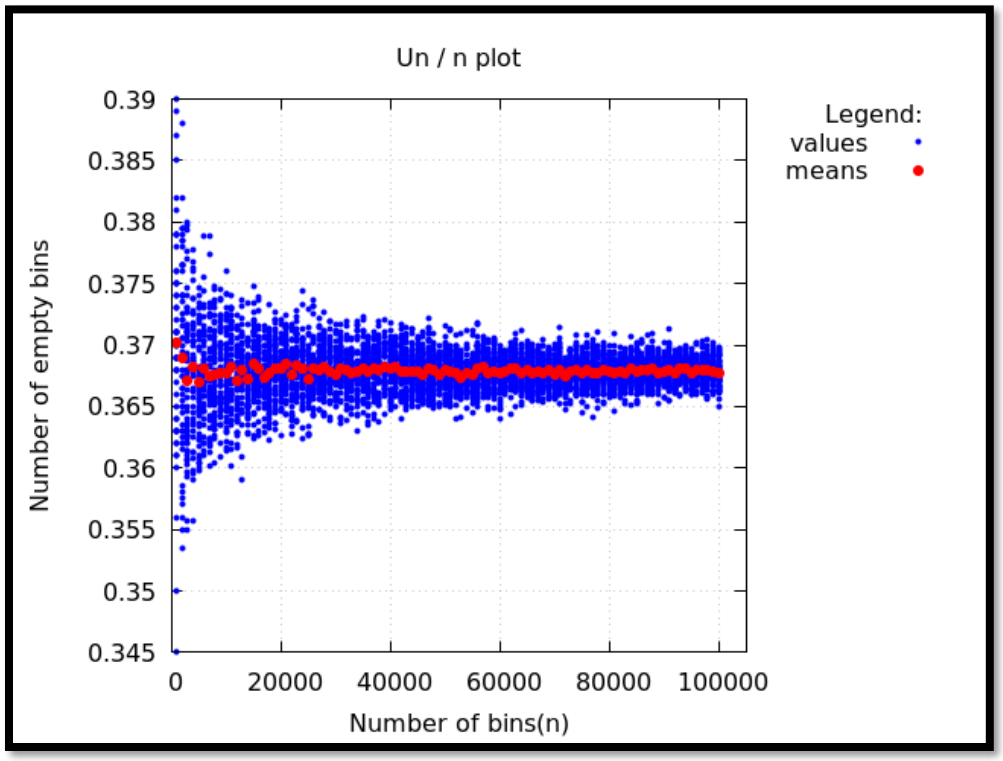
Bn pokazuje nam, że pierwsza kolizja występuje naprawdę szybko. Dla najmniejszych n zdarza się to w pierwszych <5% wrzucanych kulek, a od n ~ 20 000 jest już to <1%.

Wyniki są coraz mniej skoncentrowane wraz ze wzrostem n.

Asymptotyczne tempo wzrostu to O(sqrt(n)). Wiemy o tym ze specyfiki paradoksu urodzinowego, ale widać to też po całkiem skoncentrowanych średnich(bliskich do stałej) na wykresie Bn / sqrt(n).

Un:


Un plot
Number of empty bins
Number of bins(n)
Legend:
values
means
0
5000
10000
15000
20000
25000
30000
35000
40000
0
20000
40000
60000
80000
100000

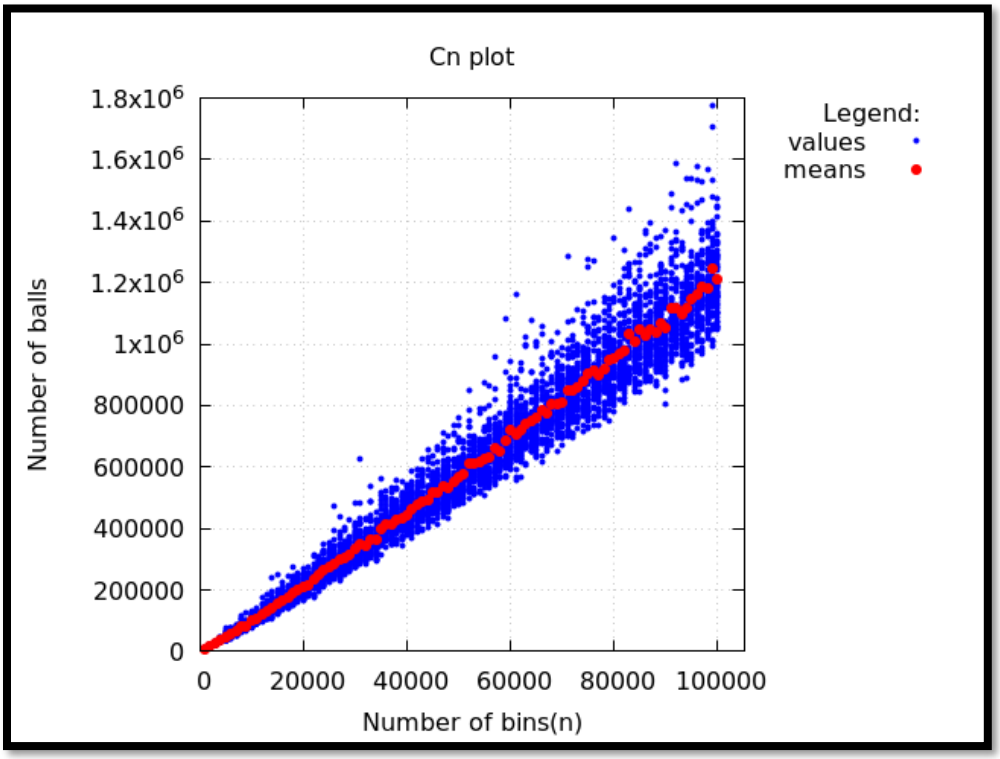

Un / n plot
Number of empty bins
Number of bins(n)
Legend:
values
means
0.345
0.35
0.355
0.36
0.365
0.37
0.375
0.38
0.385
0.39
0
20000
40000
60000
80000
100000

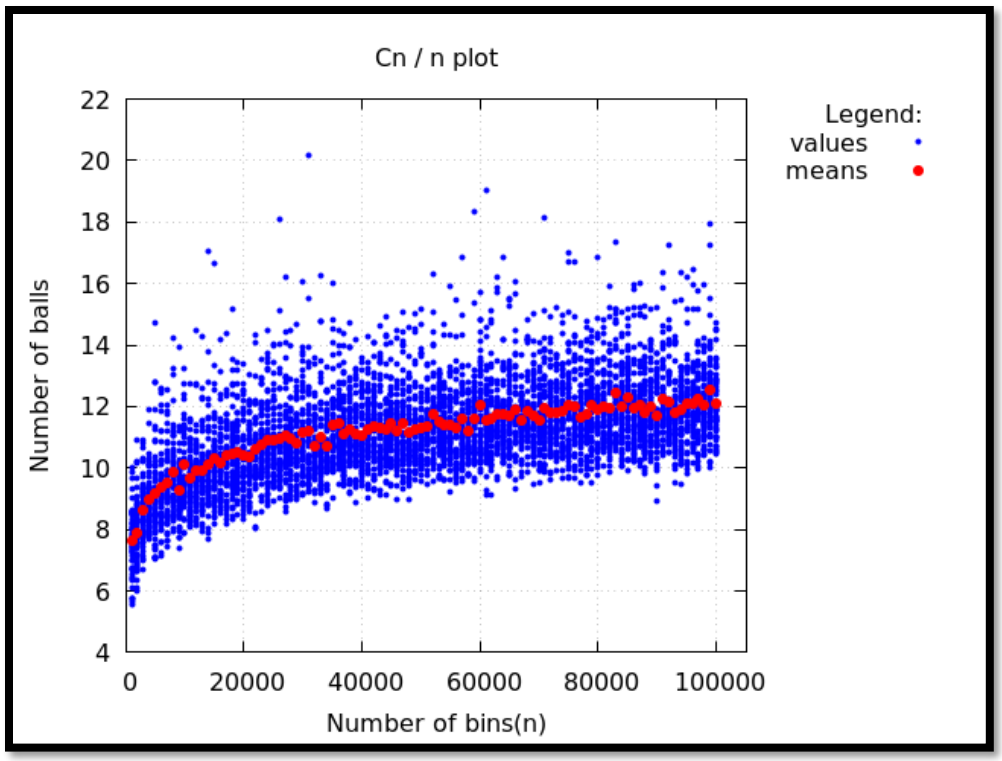
Un pokazuje nam, że stosunek pustych urn do n po wrzuceniu n kul jest stały. Pustych urn jest zawsze ~ 1/3 n. Un rośnie liniowo.

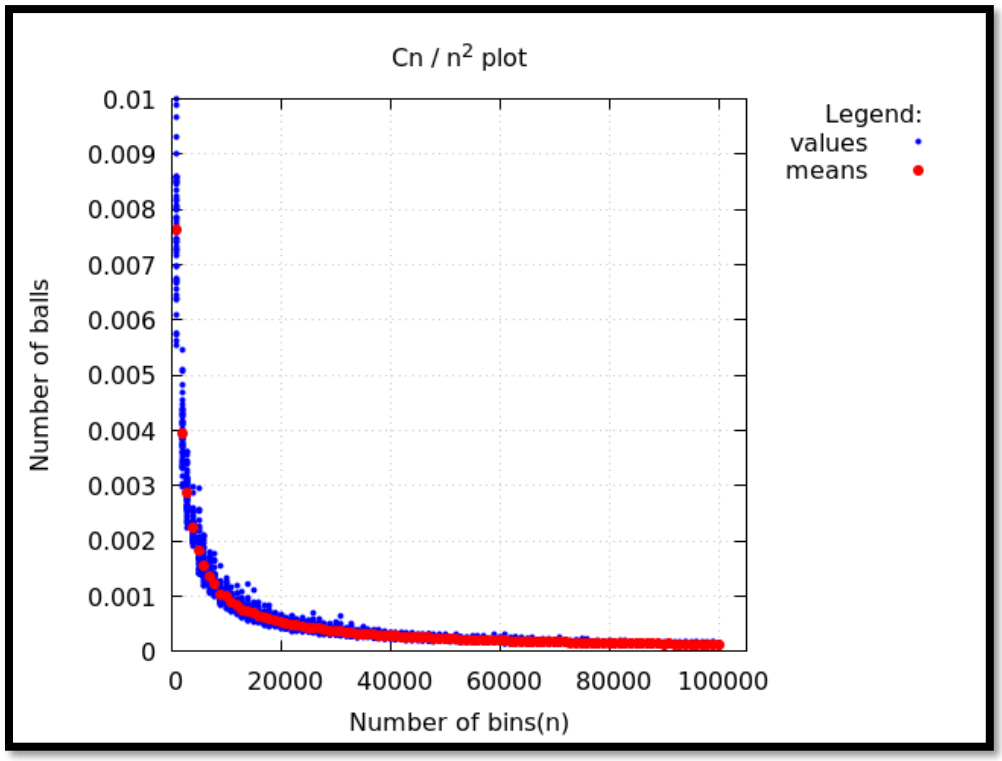
Wartości są bardzo mocno skoncentrowane, pokrywają się ze średnimi i nie występują żadne widoczne odchylenia od normy.

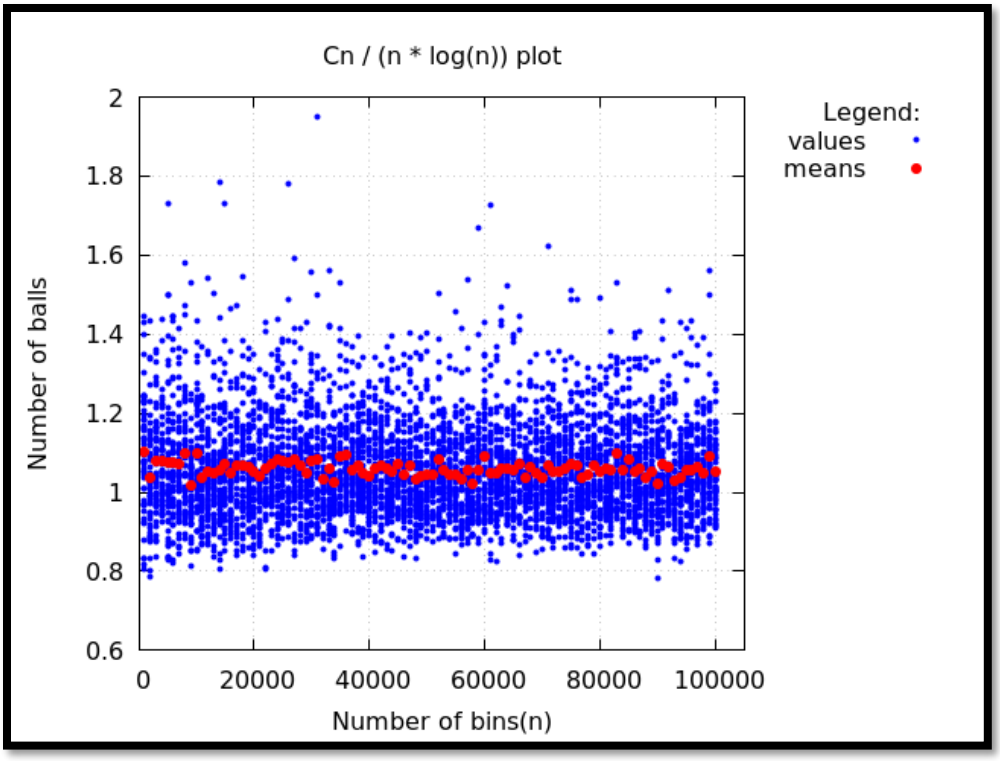
Asymptotyczne tempo wzrostu to O(n). Łatwo to tutaj dostrzec patrząc na wykres Un.

Cn:




Cn / n plot
Number of balls
Number of bins(n)
Legend:
values
means
22
20
18
16
14
12
10
8
6
4
0
20000
40000
60000
80000
100000


Cn / n2 plot
Number of balls
Number of bins(n)
Legend:
values
means
0.01
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
0
20000
40000
60000
80000
100000

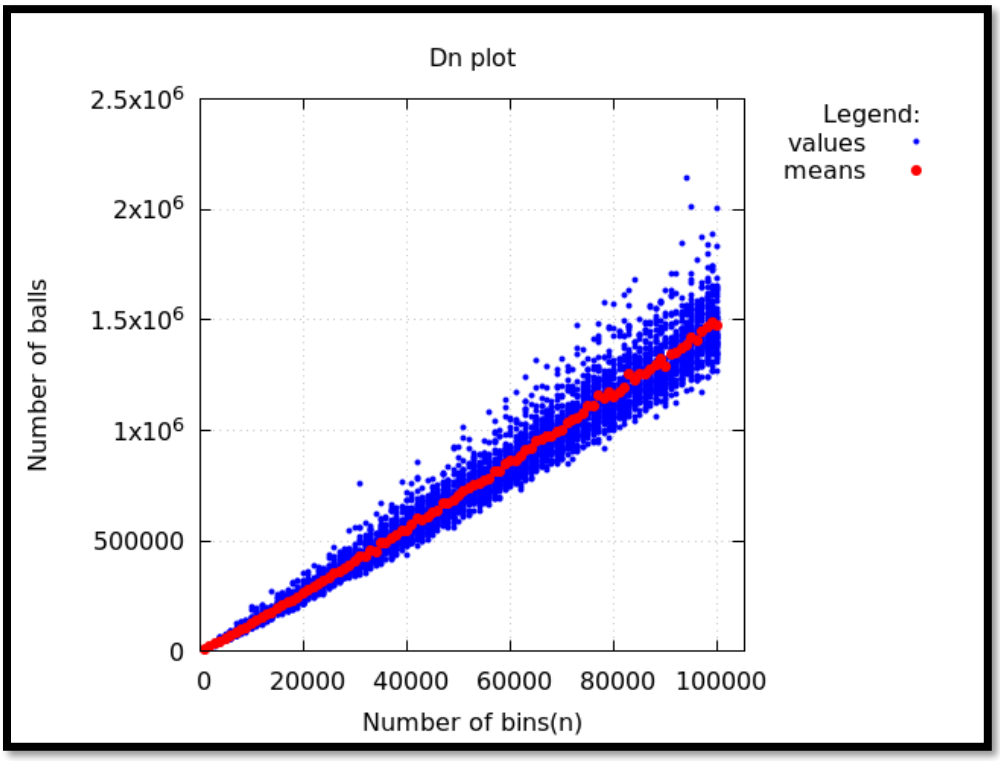


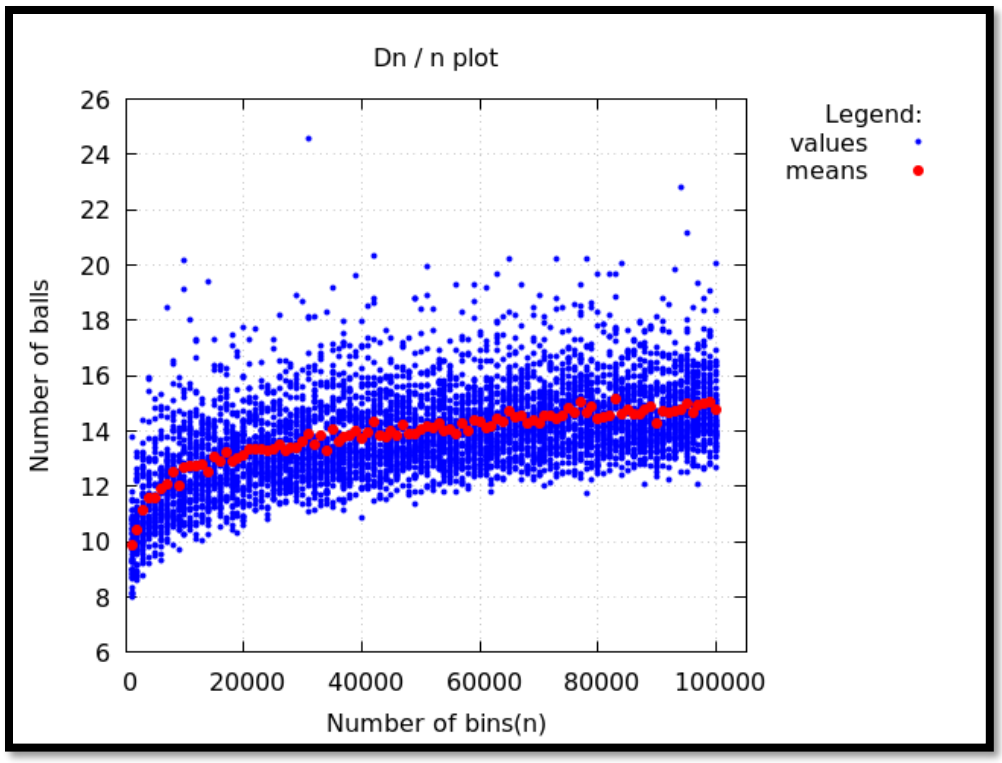
Cn pokazuje nam minimalną liczbę rzutów, po której w każdej z urn jest przynajmniej jedna kula. Ta minimalna ilość rzutów jest wysoka. Dla małych n potrzeba aż (8 do 10) * n rzutów. Od n ~ 20 000 Cn rośnie wolniej i aż do końca wykresu minimalna liczba rzutów to ~11n. Więc dla większych ilości urn, minimalna liczba rzutów rośnie coraz wolniej.

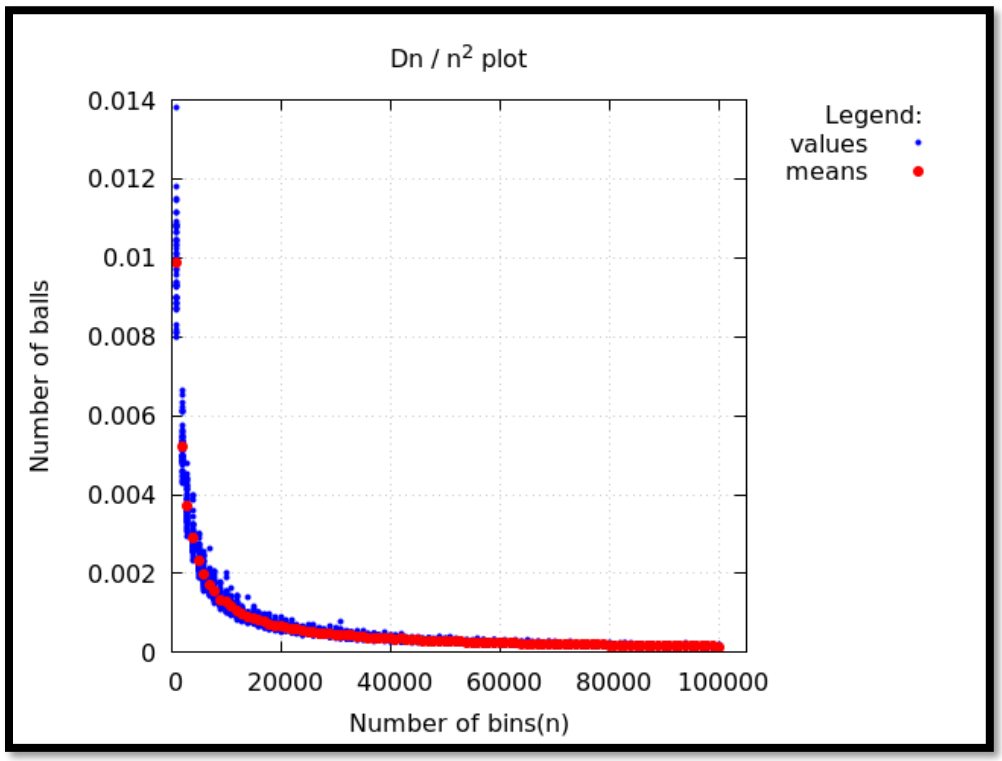
Wartości są coraz mniej skoncentrowane. Na początku koncentracja jest bardzo duża.

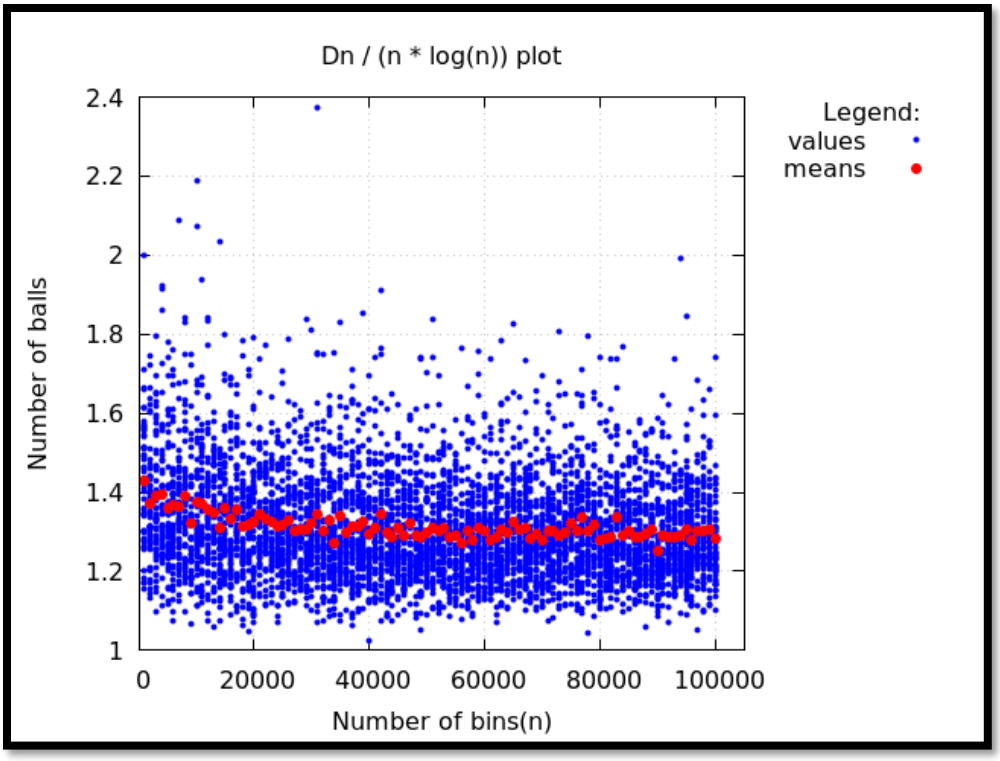
Tempo wzrostu to to O(n * log(n)). Wiemy to z problemu kolekcjonera kuponów, ale analogicznie jak wcześniej, widać to też po wykresie Cn / (n * log(n)).

Dn:


Dn plot
Number of balls
Number of bins(n)
2.5x10^6
2x10^6
1.5x10^6
1x10^6
500000
0
0
20000
40000
60000
80000
100000
Legend:
values
means


Dn / n plot
Number of balls
Number of bins(n)
Legend:
values
means
26
24
22
20
18
16
14
12
10
8
6
0
20000
40000
60000
80000
100000


Dn / n2 plot
Number of balls
Number of bins(n)
Legend:
values
means
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
20000
40000
60000
80000
100000

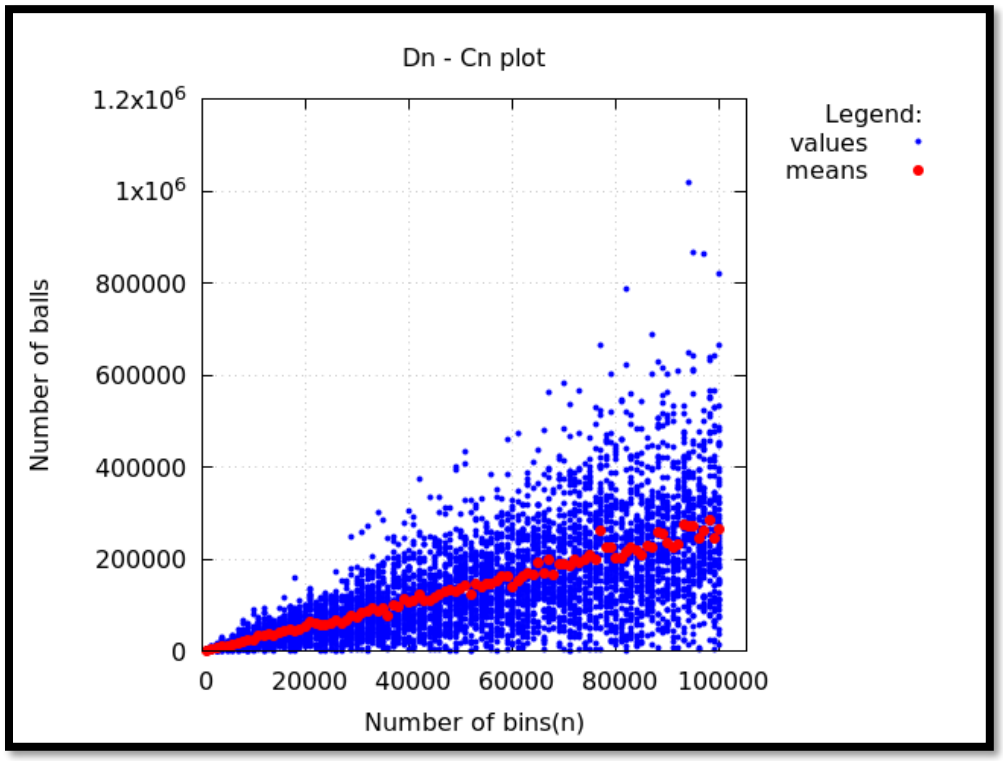


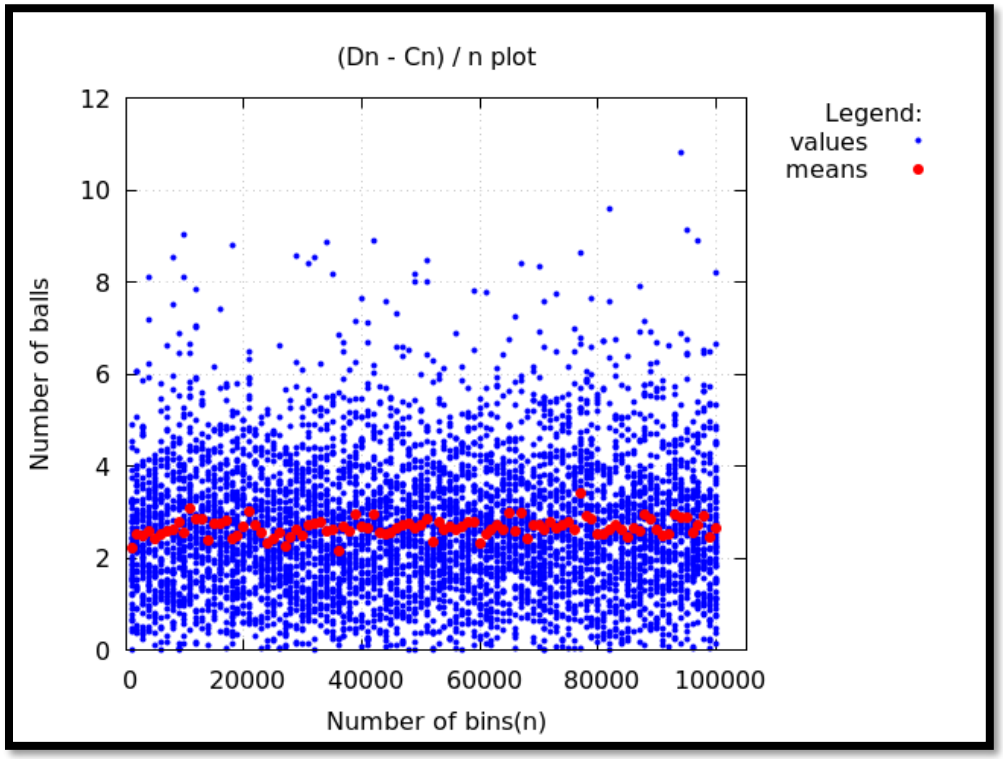
Wykres dla Dn wygląda bardzo podobnie do Cn. Dla małych n, 2 kule znajdują się w każdej urnie po wrzuceniu (10-12) * n kul. Od n ~ 20 000 wzrost jest powolniejszy i ilość kul oscyluje w okolicach 13n.

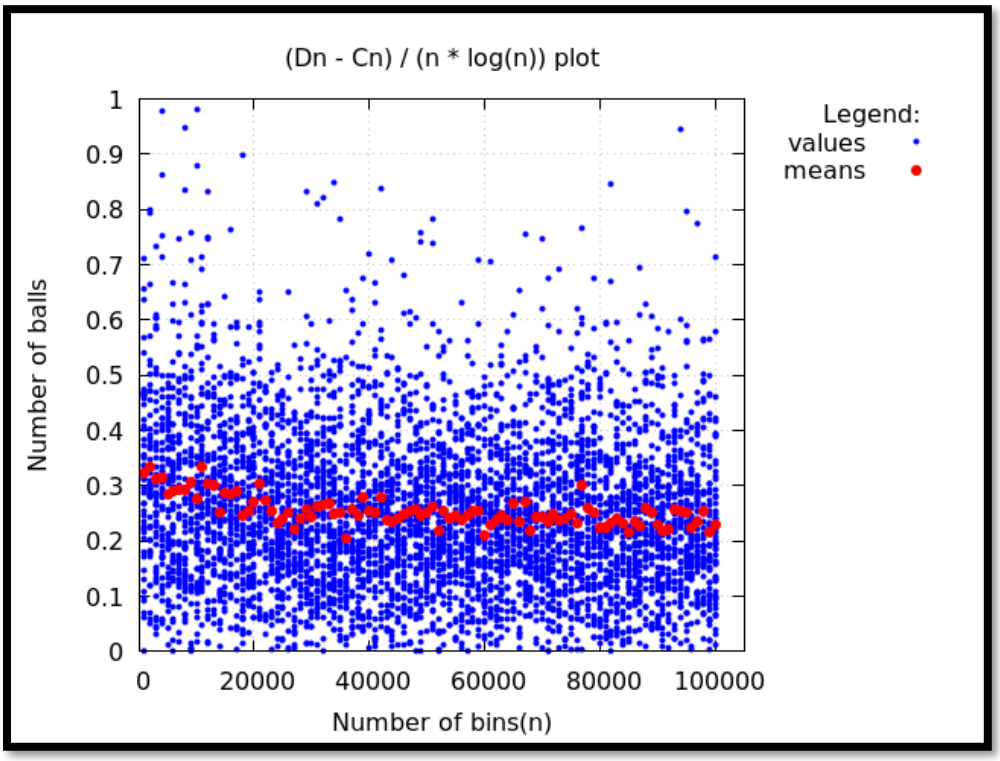
Wartości są coraz mniej skoncentrowane. Na początku koncentracja jest bardzo duża.

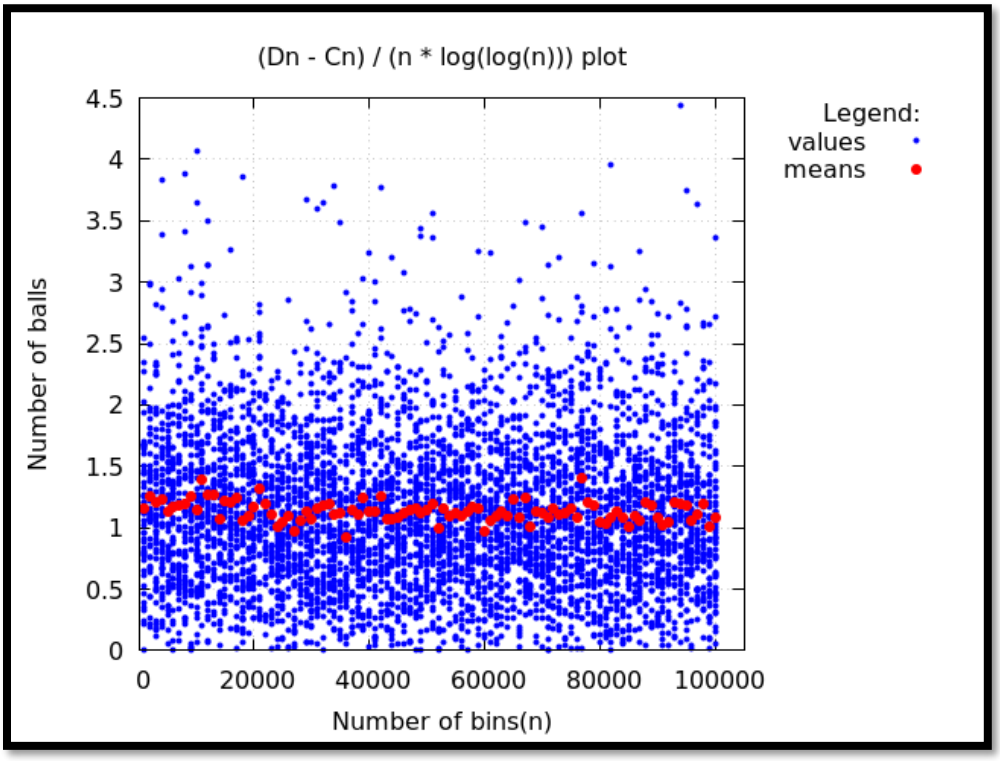
Asymptotyczne tempo wzrostu to O(n * log(n)).

Dn – Cn:


Dn - Cn plot
1.2x10^6
1x10^6
800000
600000
400000
200000
0
Number of balls
0
20000
40000
60000
80000
100000
Number of bins(n)
Legend:
values
means


(Dn - Cn) / n plot
12
10
8
6
4
2
0
Number of balls
0
20000
40000
60000
80000
100000
Number of bins(n)
Legend:
values
means





Dn – Cn pokazuje ile potrzeba rzutów aby w każdej urnie były 2 kule, po tym jak w każdej z nich znajduje się już przynajmniej jedna. Widać, że niezależnie od ilości kul potrzeba jeszcze ~2n

rzutów aby wszędzie były wrzucone 2 kule. Wydaje się to całkiem niedużo, biorąc pod uwagę jak dużo kul trzeba było wrzucić, aby wszędzie było po jednej, ale trzeba pamiętać, że w momencie wyliczenia Cn w wielu urnach może być już dużo więcej kul niż jedna.

Wartości na początku są całkiem skoncentrowane, ale szybko ich koncentracja mocno spada.

Asymptotyczne tempo wzrostu to O(n * log(log(n))).

Uzasadnienie użycia nazw *birthday paradox oraz coupon collector's problem*:

Birthday paradox:

Paradoks urodzinowy pokazuje, że znalezienie 2 osób mających urodziny tego samego dnia jest znacznie prostsze niż to by się intuicyjnie wydawało. Takich osób potrzeba zdecydowanie mniej niż jest dni w roku.

W kontekście tego zadania moment pierwszej kolizji(Bn) jest analogią do paradoksu urodzinowego. Pierwsza kolizja występuje zawsze bardzo szybko, po wrzuceniu małej ilości kulek. Więc potrzeba znacznie mniej kulek niż urn, aby doszło do kolizji(znacznie mniej ludzi, niż jest dni w roku).

W tej analogii urny to dni roku, a wrzucane kulki to ludzie mający urodziny danego dnia.

Coupon collector's problem:

Kolekcjoner kuponów stara się uzbierać komplet różnych typów kuponów z pudełek z płatkami. Po zebraniu wszystkich możliwych typów kuponów wygrywa. Pytanie brzmi, ile pudełek musi kupić, aby mieć pewność, że uzbiera wszystkie typy kuponów. Okazuje się, że jest to znacznie więcej niż mogłoby się wydawać.

W kontekście tego zadania Cn jest analogią do tego problemu. Każda urna to pudełko z kuponem innego typu. Wrzucanie kulek to kupowanie tych pudełek przez kolekcjonera. Jak widać po wynikach eksperymentu, kulek trzeba wrzucić wielokrotnie więcej, niż jest urn. I to właśnie pokazuje problem kolekcjonera kuponów.

Jakie znaczenie ma *birthday paradox* w kontekście funkcji hashujących i kryptograficznych funkcji hashujących?

W kryptografii funkcje haszujące są szeroko stosowane do jednokierunkowego przekształcania danych wejściowych na wartości wyjściowe(hashe). Algorytmy hashujące zawsze generują taki sam hash dla danego wejścia.

Paradoks urodzinowy jest używany w kryptografii do zilustrowania zagrożeń związanych ze zderzeniem haszy. Zderzenie to sytuacja, w której dwie różne wartości wejściowe mają ten sam

hasz wyjściowy. W miarę wzrostu ilości danych wejściowych (osób, w przypadku paradoksu urodzinowego), wzrasta prawdopodobieństwo wystąpienia zderzenia. Rośnie ono szybciej, niż mogłoby się nam to intuicyjnie wydawać, co wynika z paradoksu urodzinowego.

W związku z tym istnieje tak zwany *atak urodzinowy*. Wykorzystuje on paradoks urodzinowy w celu znalezienia 2 różnych zestawów danych wejściowych, które prowadzą do takiego samego wyjścia.

Z tego względu, ważne jest, aby projektować algorytmy hashujące w taki sposób, aby minimalizować ryzyko zderzeń. Jednakże, paradoks urodzinowy przypomina nam o tym, że to ryzyko zawsze będzie występować.