ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/306056778

Simple and Fast Computation of Binomial Coefficients

Conference Paper · August 2006

CITATIONS
3

READS
545

1 author:

Aleš Keprt
Moravian University College
52 PUBLICATIONS **194** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Vývoj nekonvenčních metod manažerského rozhodování v oblasti podnikové ekonomiky a veřejné ekonomie View project

All content following this page was uploaded by Aleš Keprt on 11 August 2016.

The user has requested enhancement of the downloaded file.

Simple and Fast Computation of Binomial Coefficients

Aleš Keprt

Department of Computer Science, FEI, VŠB Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava–Poruba
Department of Computer Sicence, PřF, Palacký University
Tomkova 40, 779 00 Olomouc
Ales.Keprt@vsb.cz, Ales.Keprt@upol.cz

Abstract. The paper shows a simple and quick algorithm for computation of binomial coefficients, which is needed as a supporting routine for Binary Factor Analysis.

1 Introduction

My main goal in the past few years has been the research in the field of binary factor analysis (BFA). Several BFA algorithms also utilize some combinatorial algorithms. While many combinatorial algorithms are listed in Knuth's *The Art of Computer Programming, Volume 4, Fascicle 3: Generating All Combinations and Partitions.* (Addison–Wesley, 2005), not all of what we need for BFA. The most widely used procedure is the exact computation of binomial coefficients, which is needed to let us pre–allocate the memory buffer of a correct size. This buffer is then filled with the selected subset of data row vector permutations, a.k.a. combinations, used as a starting or candidate vectors for binary factors.

All these computations are needed quite often, so they must be implemented in a computation time–effective way, while still giving the exact results.

2 Computation of Binomial Coefficients

Definition 1 (Binomial Coefficient).
The number of ways of picking k unordered outcomes from n possibilities is called ***binomial coefficient*** *or* ***combination****. It is denoted $\binom{n}{k}$ or nCk which can be also read "n choose k".*

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} \tag{1}$$

□

The binomial coefficient is needed when generating the table of all combinations (see next section) to know how much memory must be allocated. The term "picking k unordered outcomes from n possibilities" equals to "setting k bits of n-bit vector to 1s", and that's what we use in BFA computation. Namely Blind

Search BFA algorithm use the precomputed table of all $\binom{p}{i}$ combinations for some user–defined interval $i \in [min, max]$.

Direct computation from equation 1 is not possible because factorial numbers are too large to be stored in 32-bit or 64-bit integer variables. The largest factorial which fits into 32-bit unsigned integer is $13! = 479001600 \approx 0.48 \cdot 10^9$, the greatest factorial which fits into 64-bit unsigned integer is $21! = 2432902008176640000 \approx 2.4 \cdot 10^{18}$. In contrast to this, binomial coefficients are often reasonably small numbers, when k is small or similar to n. For example 28! is a huge 98-bit integer, but $\binom{28}{5} = 98280$ which fits into 17 bits.

Here we must look at what we need the binomial coefficients for. For BFA purposes, we need to allocate the memory in the size equal to the sum of successive binomial coefficients $\binom{p}{r}$ of constant p and $r \in [min, max]$. For simplicity, we can now focus to just a single binomial coefficient (i.e. $min = max$). In real applications, the factors are sparse vectors, i.e. $p >> r$. This fact corresponds to the note in the previous paragraph: The result of $\binom{p}{r}$ we need to compute can fit into 32-bit or 64-bit integer even when the base factorial $p!$ is too large to fit there. So we need a better algorithm than equation 1.

Algorithm

We start with equation 1, and cancel out $(n-k)!$.

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \frac{\prod_{i=n-k+1}^{n} i}{k!} \tag{2}$$

Since k is a reasonably small number, compared to n, and the whole fraction is a whole integer, we can scan through integers from 1 to k and cancel out each of them with one element of the fraction's numerator. Since number of numerator's and denominator's elements are equal, and they are two lists of successive integers, each denominator's element is a whole fraction of at least one numerator's element. The problem is that sometimes (for some n–k pairs), we can't easily cancel out all the denominator's elements, because two of them are cancellable with the same numerator's element, but not both at once. Since $\frac{n}{k}$ is a whole integer, we just need to express all denominator's elements as products of primes, and then they will be fully cancellable against the numerator's elements.

The following algorithm is efficient and usable for both 32-bit and 64-bit unsigned arithmetic. (I use the better 64-bit version.) Input values are in variables n and k, and the output value nCk is the requested $\frac{n}{k}$.

Since $\binom{n}{k}$ is identical to $\binom{n}{n-k}$, we at the first ensure that k is lower than $n-k$, i.e. we always take "the lower k".

```
if(n - k < k) k = n - k
```

Now we prepare the array of numerator's elements named *num*. According to formula 2 the list contains exactly k elements valued $n-k+1$ to n.

$$\forall i \in [0, k-1] : num_i = n - i$$

Now we cancel out each of k denominator's elements valued 1 to k using this tiny loop.

```
For each i in interval [2,k] {
  while i is nonzero and divisible by 2 {
    divide i by 2
    cancel out 2 from numerators
  }
  while i is nonzero and divisible by 3 {
    divide i by 3
    cancel out 3 from numerators
  }
  while i is nonzero and divisible by 5 {
    divide i by 5
    cancel out 5 from numerators
  }
  if i > 1 {
    cancel i from numerators
  }
}
```

We repeat the inner `while` loops only for primes $2, 3, 5$, because that's exactly enough for all up to 64-bit unsigned integers. The reason will be discussed below.

Now we have cancelled out all denominator's elements. The product of the resulting numerators elements in *num* array is the wanted $\binom{n}{k}$.

$$\binom{n}{k} = \sum_{i=0}^{k-1} num_i \tag{3}$$

Implementation of this equation must ensure that no arithmetic overflow occur. We can do it as follows:

```
nCk = 1 for each i in interval [0,k-1] {
  if nCk * num[i] / nCk != num[i] {
    //error - arithmetic overflow
  }
  nCk *= num[i]
}
```

We use a simple way of arithmetic overflow checking: $A * B$ overflows if and only if $A * B/A \neq B$. If no overflow occurs, variable nCk contains the wanted $\binom{n}{k}$.

The proofs

The correct functioning of the above algorithm is directly distinct from its description. We just need to prove two theorems.

Theorem 1.
In order to compute up to 64-bit $\binom{n}{k}$ *values, the inner* `while` *loops only need to be performed for primes* $2, 3, 5$*.*

Proof. The primes are used to express denominator's elements. The greatest of them is k, so it's the greatest value to be expressed as the product of primes. If we look at the 64-bit $\binom{n}{k}$ values, the ones with the greatest k are $\binom{66}{33}$ and $\binom{67}{33}$, what was empirically observed, so the greatest k of our interest is 33.

Since $k = \sqrt{k} \cdot \sqrt{k}$, each integer k can be expressed as a product of primes from whose at most one is greater than $\sqrt{k}$. The other primes are always less than or equal to $\sqrt{k}$. Since we are interested in numbers no greater than 33, and $\sqrt{33} \approx 5.7$, they all can be expressed by primes 2, 3, 5, and at most one prime greater than 5. This one last possible prime is checked after all three `while` loops (see the algorithm above). $\square$

Theorem 2.
To detect an arithmetic overflow when multiplying two unsigned integers A *and* B*, we just need to test whether* $A * B/A = B$*. In the case of overflow, this equality doesn't apply.*

Proof. Obviously, in the real world the equality $A*B/A = B$ always apply. Let's mark the correct result of $A*B$ as C. In computer world, the integers are limited by the number of bits used for their representation. Let's mark T the greatest integer that can fit into our representation. If $C > T$, then $A * B$ can't fit into these bits, and the resulting product is smaller than it should be, i.e. $A * B < C$. Since $C/A = B$ and $A * B < C$, also $A * B/A < B$. And that's the wanted proof.

One last thing we need to do is to take care of integer–rounding. If C and $A * B$ were close enough, then an equality $C/A = A * B/A$ could occur even in the case of overflow, and lead us to miss this overflow. Fortunately, this situation can't occur, as C and $A*B$ are never close to each other, since when an overflow occurs, and C is close to T, the resulting $A * B$ is close to 0.

The conclusion is that in the case of arithmetic overflow, the inequality $A * B/A < B$ applies. $\square$

Publications in technical journals

1. Keprt, A.: Horké novinky v jazyce C# 2.0. (3 parts). In: *ComputerWorld.* Volume **15**. No. 42,43,44/2004. ISSN 1210-9924.
2. Keprt, A.: Historie DirectX. In: *ComputerWorld.* Volume **11**. No. 28/2000. ISSN 1210-9924.
3. Keprt, A.: Rozhraní DirectInput (2 parts). In: *ComputerWorld.* Volume **11**. No. 27,28/2000. ISSN 1210-9924.

4. Keprt, A.: Jak si popovídat se svým PC – DirectSoundCapture. In: *ComputerWorld.* Volume **11**. No. 26/2000. ISSN 1210-9924.
5. Keprt, A.: Dovolte, abych se představil, mé jméno je DirectX. In: *ComputerWorld.* Volume **11**, No. 14/2000. ISSN 1210-9924.
6. Keprt, A.: Textové editory pro ZX Spectrum (2 parts). In: *ZX Magazín.* **1994**, Vol. 1994, No. 1,2/1994. ISSN 1210-4833.

Publications in conference proceedings – foreign events

7. Húsek, D., Keprt, A., Řezanková, H., Frolov, A.A., Polyakov, P., Snášel, V.: Comparison of Different Approaches to Factorization of Binary Variables. In proceedings of *ITAT 2005,* Račkova Dolina, Slovakia. Ed. Peter Vojtáš, Univerzita Pavla Jozefa Šafárika, Košice, Slovakia, **2005**, pp. 55–64, ISBN 80-7097-609-8.
8. Keprt, A.: Paralelní řešení binární faktorové analýzy. In proceedings of *ITAT 2005,* Račkova Dolina, Slovakia. Ed. Peter Vojtáš, Univerzita Pavla Jozefa Šafárika, Košice, Slovakia, **2005**, pp. 223–232, ISBN 80-7097-609-8.
9. Keprt, A., Snášel, V.: Binary Factor Analysis with Genetic Algorithms. In proceedings of *4th IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology – WSTST 2005.* Springer Verlag, Berlin Heidelberg, Germany, **2005**, pp. 1259–1268, ISBN 3-540-25055-7. In series Advances in Soft Computing, ISSN 1615-3871.
10. Keprt, A., Snášel, V.: Pseudo-dělení binárních matic a jeho aplikace. In proceedings of *Znalosti 2005.* Vysoké Tatry, Slovakia. Published by VŠB Technická Univerzita, Ostrava, **2005**, pp. 41–50, ISBN 80-248-0755-6.
11. Keprt, A., Snášel, V.: Genetické algoritmy pro redukci dimenze a analýzu binárních dat. In proceedings of *Znalosti 2005.* Vysoké Tatry, Slovakia. Published by VŠB Technická Univerzita, Ostrava, **2005**, pp. 242–249, ISBN 80-248-0755-6.
12. Keprt, A., Fojták, M.: Komprese barevného obrazu vícevrstvou neuronovou sítí. In poster proceedings of *Znalosti 2005.* Vysoké Tatry, Slovakia. Published by VŠB Technická Univerzita, Ostrava, **2005**, pp. 53–56.
13. Keprt, A., Zlý, M.: Efektivní implementace neuronových sítí pomocí vektorových instrukcí. In poster proceedings of *Znalosti 2005.*, Vysoké Tatry, Slovakia. Published by VŠB Technická Univerzita, Ostrava, **2005**, pp. 57–60.
14. Keprt, A., Snášel, V.: Binární faktorová analýza genetickým algoritmem. In proceedings of *ITAT 2004.* Vysoké Tatry, Slovakia, **2005**, 10 pp.
15. Keprt, A., Zlý, M.: Využití SIMD instrukcí pro implementaci neuronových sítí. In proceedings of *ITAT 2004.* Vysoké Tatry, Slovakia, **2005**, 10 pp.

Chapters in books

16. Lacko, B. & Ševčík, V. (editors): *Kybernetika a společnost na prahu XXI. století.* Chapter XI. Vzory v návrhu her. VUT Brno, **2005**, pp. 59–64, ISBN 80-214-3058-3.

Publications in conference proceedings – Czech events

17. Keprt, A.: Possible Interpretations of Binary Factor Analysis. In proceedings of *Znalosti 2006,* Hradec Králové. VŠB Technical University, Ostrava, Czech Republic, **2006**, pp. 280–283, ISBN 80-248-1001-8.

18. Keprt, A.: Thread Local Storage. In proceedings of *Objekty 2005.* VŠB Technical University, Ostrava, Czech Republic, **2005**, pp. 85–91, ISBN 80-248-0595-2.
19. Keprt, A.: Recent Advances in Binary Factor Analysis. In proceedings of *Wofex 2005.* VŠB Technical University, Ostrava, Czech Republic, **2005**, pp. 376–381, ISBN 80-248-0866-8.
20. Keprt, A.: Vzory v návrhu her. In proceedings of *Tvorba softwaru 2005.* Tanger s.r.o., Ostrava, **2004**, pp. 61–68, ISBN 80-86840-14-X.
21. Keprt, A.: Digitalizace časopisu Falstaff. In proceedings of *Tvorba softwaru 2005.* Tanger s.r.o., Ostrava, **2004**, pp. 69–75, ISBN 80-86840-14-X.
22. Keprt, A.: Nulovatelné typy pod lupou. In proceedings of *Tvorba softwaru 2005.* Tanger s.r.o., Ostrava, **2004**, pp. 76–82, ISBN 80-86840-14-X.
23. Keprt, A.: Nové prvky jazyka Visual C# 2.0 (2005). In proceedings of *Objekty 2004.* Ed. David Ježek, Vojtěch Merunka, VŠB Technická Univerzita, Ostrava, **2004**, pp. 15–32, ISBN 80-248-0672-X.
24. Keprt, A.: Vývoj a řízení projektu síťové počítačové hry. In proceedings of *Objekty 2004.* Ed. David Ježek, Vojtěch Merunka, VŠB Technická Univerzita, Ostrava, **2004**, pp. 98–108, ISBN 80-248-0672-X.
25. Keprt, A., Snášel, V.: Binary Factor Analysis with Help of Formal Concepts. In proceedings of *CLA / Concept Lattices and their Applications.* Ed. Václav Snášel, Radim Bělohlávek, VŠB Technická Univerzita Ostrava, Czech Republic, **2004**, pp. 90–101, ISBN 80-248-0597-9.
26. Keprt, A.: Binary Factor Analysis. In proceedings of *Wofex 2004.* VŠB Technická Univerzita, Ostrava, Czech Republic **2004**, pp. 298–303, ISBN 80-248-0596-0.
27. Keprt, A.: Jazyk C# a .NET Framework na Linuxu. In proceedings of *Tvorba softwaru 2004.* Tanger s.r.o., Ostrava, **2004**, pp. 97–105, ISBN 80-85988-96-8.
28. Keprt, A.: Binary Factor Analysis and Its Usage in Data Mining. In proceedings of *Poster 2004.* ČVUT (České vysoké učení technické), Fakulta elektrotechnická, Praha, **2004**.
29. Keprt, A.: Using Blind Search and Formal Concepts for Binary Factor Analysis. In proceedings of *Dateso 2004.* Ed. Václav Snášel, Jaroslav Pokorný, Karel Richta, VŠB Technická Univerzita Ostrava, Czech Republic; CEUR WS – Deutsche Bibliothek, Aachen, Germany; **2004**, pp. 120–131, ISBN 80-248-0457-3 (VŠB TUO), ISSN 1613-0073 (CEUR).
30. Húsek, D., Frolov, A.A., Keprt, A., Řezanková, H., Snášel, V.: O jednom neuronovém přístupu k redukci dimenze. In proceedings of *Znalosti 2004.* VŠB Technická Univerzita, Ostrava, **2004**, pp. 327–337, ISBN 80-248-0456-5.
31. Keprt, A.: PCA a porovnávání zkompresovaných obrázků. In poster proceedings of *Znalosti 2004.* Ed. Václav Snášel, VŠB Technická Univerzita, Ostrava, **2004**, pp. 29–32.
32. Keprt, A.: Architektura DirectShow. In proceedings of *Objekty 2003.* VŠB Technická Univerzita, Ostrava, **2003**, pp. 108–119, ISBN 80-248-0274-0.
33. Keprt, A.: Binární faktorová analýza a komprese obrazu pomocí neuronových sítí. In proceedings of *Wofex 2003*, VŠB Technická Univerzita, Ostrava, **2003**, ISBN 80-248-0106-x.
34. Keprt, A.: DirectX. In proceedings of *Objekty 1999.* Ed. Vojtěch Merunka, Vladimír Sklenář, Česká Zemědělská Univerzita, Praha, **1999**, pp. 215–222, ISBN 80-213-0552-5.

View publication stats