Restricted Partitions

An NP-Complete and Number Theory Problem

What Are Partitions?

- A partition of some number is a sum of positive numbers called parts with the same value as the original number.
- Unrestricted partitions include all combinations.
- Example: The partitions of 5 are

5	2+2+1
4+1	2+1+1+1
3+2	1+1+1+1+1
3+1+1	

What Are Restricted Partitions?

- Restricted partitions require that valid partitions of a number follow certain rules.
- These rules could be something like all parts must be odd or all parts must be distinct.
- Multiple restrictions can be used.
- Example: The partitions of 5 with odd parts are
  - 5
  - 3+1+1
  - 1+1+1+1+1

Why Is This Problem Hard?

- Counting the number of partitions is difficult because it is an NP-Complete problem.
- “NP” stands for non-deterministic polynomial time (as a general rule, much longer than you want to wait, like many days or years).
- An NP problem can be easily verified, but it takes a lot of effort to generate solutions.
- “A lot of effort” means that even small problems can take many years of straight computer time to solve.

Proving NP-Completeness

- It is easy to make a problem seem harder than it actually is and to make a problem NP-Complete when it really isn't.
- There are two parts to an NP-Complete proof:
  - Prove that the problem is in the set NP, which means that its solution can be verified in polynomial time (a reasonable amount of time that doesn't increase exponentially).
  - The problem can be reduced to another NP-Complete problem.

Basis of NP-Complete Proof

- First, is a solution verifiable in polynomial time?
- Is 1+2+7+12 a partition of 22?
- To solve this, we add up all elements in our solution set {1,2,7,12} to get 22.
- This can be done in linear time because it is directly related to the number of elements in the set.
- Linear time is polynomial time, so the problem is a member of the set NP.

Basis of NP-Complete Proof

- Can we reduce a problem from the known NP-Complete catalog to the Partition problem?
- One problem is the Subset Sum problem, which is a relationship that is true if a subset of a given set of numbers adds up to a given *n*, but is false otherwise.

$\{-4, 0, 3, 9, 15, 18\}$; $n = 5$

- {-4,9} is the only subset whose sum is 5.

Basis of NP-Complete Proof

- Let us define the set A as available numbers as
  $$A = \{a_1, a_2, \dots, a_k\}\ where\ a_1 \leq a_2 \leq \cdots \leq a_k,$$
- We can modify this problem without changing its solution set by adding $(-a_1 + 1)$ to every term in A and adding $(-a_1 + 1)x$ to n where x is the number of elements used to construct a subset sum.

$\{-4+5, 0+5, 3+5, 9+5, 15+5, 18+5\}$; $n = 5+5x$

$\{1, 5, 8, 14, 20, 23\}$; $n = 5+5x$

Basis of NP-Complete Proof

$\{1, 5, 8, 14, 20, 23\}$; $n = 5+5x$

- As shown, there are no longer any non-positive numbers in the set.
- If we still use the newly modified elements that yielded solution before, x=2, and, therefore, n=15.
- Since 1+14=15, the elements we were interested in yield a solution before and after the transformation.
- This transformation can be done in linear time, so it fits all of our criteria.

Basis of NP-Complete Proof

- We successfully transformed the Subset Sum problem into a Partition problem where all parts are distinct.
- To model the Subset Sum problem as an Partition problem with different or no restrictions, we can vary restrictions on Subset Sum set elements.
- Unrestricted – allow multiple uses of elements
- Evens only – only even elements
- All variations of the Subset Sum problem have been proven to be NP-Complete.

Restricted Partition $h(m,n)$

- The restricted partition studied was $h(m,n)$
- $h(m,n)$ represents the total number of partitions of the number n where the smallest part is exactly m .
- $h(m,n)$ can be expanded into the difference of two simpler function calls through the use of a recurrence relation.

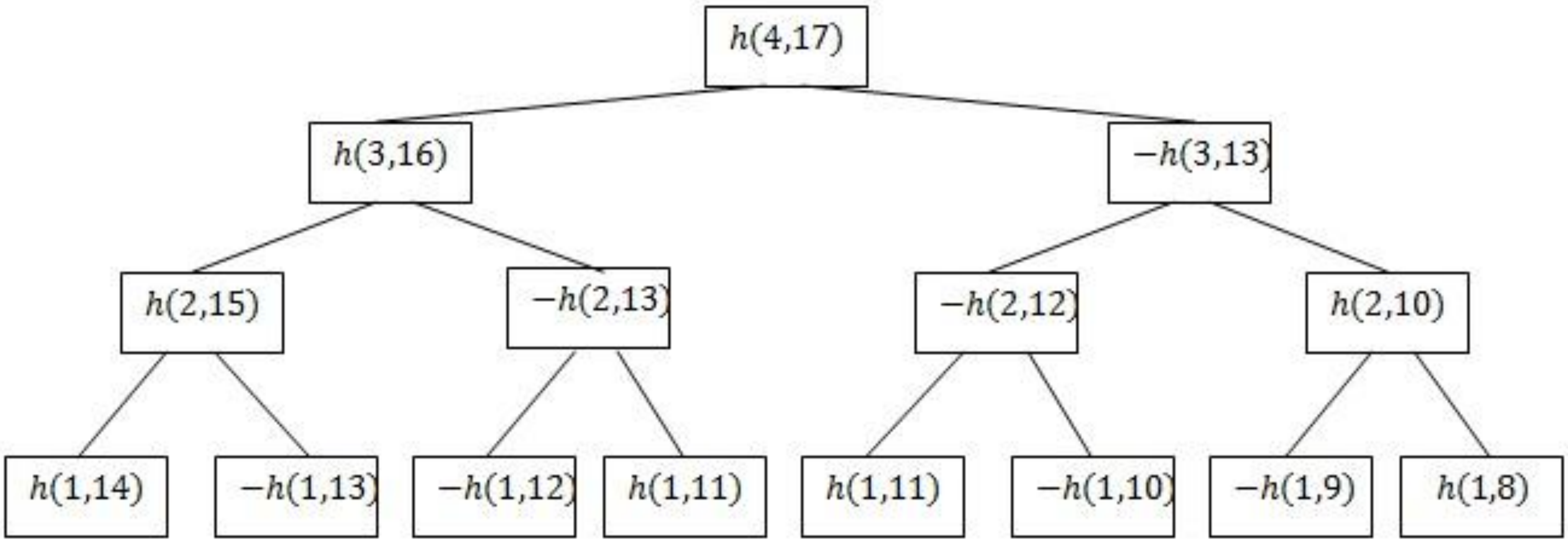
Restricted Partition $h(m,n)$

- The recurrence relation

$$h(m,n) = h(m-1,n-1) - h(n-1,n-m)$$

  can be used to decompose the function $h(m,n)$ into a sum of functions with smaller m and n values.
- This is done until the original $h(m,n)$ function is represented as a sum of functions in the form $h(1,x)$.
- This will simplify counting the partitions.

Decomposition Example



The function $h(4,17)$ being decomposed into a sum of the simplest functions possible shown in a binary tree format.

Sign Patterns

- The simplified function calls exhibit a pattern that can be used to generalize the calculations.
- The signs of the first 64 function calls are shown below from left to right in blocks of 16:

+--+	-++-	-++-	+--+
-++-	+--+	+--+	-++-
-++-	+--+	+--+	-++-
+--+	-++-	-++-	+--+

+ represents positive; – represents negative

Sign Patterns

- The sign for the k^{th} function call is represented by
$$f(k) = \begin{cases} 1, where\ the\ leaf\ at\ position\ k\ is\ positive \\ -1, where\ the\ leaf\ at\ position\ k\ is\ negative \end{cases}$$
- This function is defined by nested recursive applications of the sign pattern to a group of function calls whose size is a power of 4.
- The pattern is $X\overline{X}\,\overline{X}X$ where X represents a pattern of signs for 4^k function calls where k is a non-negative integer. $\overline{X}$ represents a pattern of signs for 4^k function calls where all positive signs in X are negative and vice-versa.

Sign Pattern Examples

- Where $k = 0$:
  - X = +
  - $\overline{X}$ = -
  - $X\overline{X}\,\overline{X}X$ = +--+
- Where $k = 1$
  - X = +--+
  - $\overline{X}$ = -++-
  - $X\overline{X}\,\overline{X}X$=+--+-++--++-+--+

Sign Pattern Examples

- Where k = 2:

X	$\overline{X}$	$\overline{X}$	X
+--+	-++-	-++-	+--+
-++-	+--+	+--+	-++-
-++-	+--+	+--+	-++-
+--+	-++-	-++-	+--+

- Note that, when grouped in blocks of 16, the pattern for k = 2 is the same as the pattern for k = 0 where the "X" pattern of plusses replaces "+" and the "O" pattern of plusses replaces "-".

Second Argument Patterns

- The second argument of decomposed function calls in the form $h(1, x)$ also follow a pattern.
- When arranged in blocks of 16 as shown, the terms in the positions of the bold numbers cancel (Example shown for $h(16{,}600)$.)
- 8 of every 16 terms cancel, halving computation time.

$$h(1{,}585) - h(1{,}584) - h(1{,}583) + \boldsymbol{h}(\mathbf{1}, \mathbf{582})$$
$$-\boldsymbol{h}(\mathbf{1}, \mathbf{582}) + \boldsymbol{h}(\mathbf{1}, \mathbf{581}) + h(1{,}580) - \boldsymbol{h}(\mathbf{1}, \mathbf{579})$$
$$-\boldsymbol{h}(\mathbf{1}, \mathbf{581}) + h(1{,}580) + \boldsymbol{h}(\mathbf{1}, \mathbf{579}) - \boldsymbol{h}(\mathbf{1}, \mathbf{578})$$
$$+\boldsymbol{h}(\mathbf{1}, \mathbf{578}) - h(1{,}577) - h(1{,}576) + h(1{,}575)$$

Decrement Function

- It is possible to generate the second argument for the k^{th} $h(1,x)$ term using only the least part m, the number to partition n, and k, the index of the term.
- The function $d(k)$ returns the total decrement from the original term $x(0)$.
- Let

$$x(k) = x(0) - d(k)$$

- Using $d(k)$, we can find the k^{th} $h(1,x)$ term without using recursion.

Decrement Function

- Write *k* as a linear combination
$$k = c_0 4^0 + c_1 4^1 + \cdots + c_q 4^q.$$
and
$$k = \sum_{j=0}^{p} c_j 4^j \; c_j \in \{0,1,2,3\} \; where \; p = [\log_4 k].$$
where [*x*] is defined as the largest integer that is at most *x*.
- Each summand in the linear combination will be used to find part of the entire *d(k)*.

Decrement Function

- *d(k)* uses recursion to generate a sum of partial decrements starting from the largest power of 4.
- We define $g(c_p,p)$, a function that generates partial decrements, as

$$g(c_p, p) = \begin{cases} 2p + 1, c_p = 1 \\ (2p + 1) + 1,\ c_p = 2 \\ 2(2p + 1) + 1,\ c_p = 3 \end{cases}$$

where c_p is the coefficient, and p is the power.

Decrement Function

- Using the definitions for k and $g(c_p,p)$, we define $d(k)$ recursively as

$$
\begin{aligned}
d(0) &= 0\\
d(1) &= 1\\
d(k) &= g\left(c_q, q\right) + d\left(k - c_q 4^q\right).
\end{aligned}
$$

- An example of $d(14)$ is provided in the next slide.

Decrement Function

- *(Add d(k) example based on h(16,600), which had its first 16 terms given on slide 18)*

Unrestricted Partitions vs *h(1,n)*

- $h(1, n+1) = p(n)$
- As shown below, each partition of *n* with least part of 1 is equivalent to an unrestricted partition of *n*-1 with a 1 prepended to the sum.

n	Partitions for $p(n)$	$p(n)$	Partitions for $h(1,n)$	$h(1,n)$
1	1	1	1	1
2	1+1;2	2	1+1	1
3	1+1+1;1+2;3	3	1+2;1+1+1	2
4	1+1+1+1;1+1+2;1+3;2+2;4	5	1+1+1+1;1+1+2;1+3	3
5	1+1+1+1+1;1+1+1+2;1+2+2;1+1+3; 2+3;1+4;5	7	1+1+1+1+1; 1+1+1+2;1+1+3;1+4; 1+2+2	5
6	1+1+1+1+1+1;1+1+1+1+2;1+1+2+2; 2+2+2;1+1+1+3;1+2+3;3+3;1+1+4;2+4; 1+5;6	11	1+1+1+1+1+1; 1+1+1+1+2; 1+1+1+3;1+1+4;1+5; 1+2+3;1+1+2+2	7