SONGS AND THE ANALYSIS OF ALGORITHMS

Darrah Chavey
Department of Mathematics and Computer Science
Beloit College
700 College St; Beloit, WI 53511
chavey@beloit.edu

ABSTRACT

In teaching a course that includes the analysis of algorithms, songs can be used as teaching examples. Multiple examples can be done at one time, or in one homework set, without the background explanation required to analyze real algorithms. Many components of algorithm analysis can be introduced using an analysis of songs. This is particularly useful at gaining experience extracting summations from "word problems."

1. OVERVIEW

In teaching analysis of algorithms, we would like to present multiple examples to help students understand the basic, common elements. Unfortunately, demonstrating or assigning several summations or recurrence relations is often too repetitive—e.g. there are a limited number of ways to disguise $\sum_{i=1}^{n} i$. Analysis of real, interesting algorithms is preferred, but to do so we need to spend significant time describing the algorithm before we can get to the analysis. This approach dilutes the introduction of "analysis" across several weeks. We would like to concentrate, for a portion of the course, on the techniques of analysis alone. One partial solution is to analyze common folk songs in an introductory lecture (and homework assignment) for this material. Here we ask, "How long would it take to sing n verses of this song?" For most songs, the answer is $\bigcirc(n)$, but for many folk songs and children's songs the answer is more interesting.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise or to republish, requires a fee and/or specific permission.
SIGCSE '96 2/96 Philadelphia, PA USA
©1996 ACM 0-89791-757-X/96/0002....$3.50

To be more precise, in this paper we define $T(n) =$ *the number of syllables sung while singing n verses of this song.* We often have to be imaginative in deciding what "new" verses would be like, as in *The N Days of Christmas* when $N > 12$, but we assume that this can be done in a manner consistent with the existing verses. This definition in terms of syllables avoids certain problems associated with the metrical qualities of songs, implies that the songs can be analyzed without reproducing the tune, and allows us to be more precise about the constants in the analysis (if we wish to be).

2. A FIRST CLASSROOM EXAMPLE

To demonstrate how we might use this device in a class, we consider the song *I Know An Old Lady Who Swallowed a Fly*, which is a reasonable first classroom example. We give more details about the lyrics in our class, and later in this paper, but indicate the general features here. In this song, the traditional first 7 verses have size given by:

$$T(1) = 24$$
$$T(2) = T(1) + 23$$
$$T(3) = T(2) + 17$$
$$T(4) = T(3) + 18$$
$$T(5) = T(4) + 17$$
$$T(6) = T(5) + 19$$
$$T(7) = T(6) + 18$$

In class we point out that this is similar to the behavior of a simple loop where each iteration of the loop takes a (nearly) constant amount of time. As with true algorithms, it is clear that there are trivial differences between iterations that make no substantial difference to the analysis, and we lose no essential information when we replace the equations above by:

$$T(1) = 24; \quad T(n) = T(n-1) + 18$$

After solving this via a simple summation, we then analyze a similar song, such as *Alouette*, which satis-

fies (to the same close approximation) the relations:

$$T(1) = 40; \quad T(n) = T(n-1) + 7$$

These two examples might take 20 minutes, less than analyzing two sorting algorithms. Nevertheless, we can be much more precise about the summation formula we develop than we could in this time with a sorting algorithm. The summations we develop are the same as those we will need for "real" algorithms. We can easily discuss the relative importance of the constants, and the strengths and weaknesses of $\bigcirc()$ notation. For discussion of $\bigcirc()$ notation, these examples are better than using n^2 sorting algorithms, whose n^2 coefficients are quite similar. This is simply not true for more complicated algorithms, and it is also not true for these songs. For example, the n^2 coefficient of the first example is 9, while that of the second example is 3.5, and the examples of section 4 have coefficients from 3 to 11. (The difference in leading coefficients is even more dramatic with the more complicated examples of that section.)

We can also demonstrate very effectively how the "leading term" can be less significant than expected for small values of n. For example, *She'll Be Coming 'Round the Mountain* is $\bigcirc(n^2)$, but the $\bigcirc(n)$ term dominates up to $n = 53$ (small for computer work, but massive for singing). The difference in dominant terms is easy to hear if you're willing to play a tape of someone singing, say, verse 1 and 10 of this song and compare it to a tape of a song with a more dominant $\bigcirc(n^2)$ term. This gives the student a much more concrete sense of "dominant term" than we can get in even a few weeks of work with "standard" algorithms.

At this point in the course, it is easy to assign the analysis of one or more songs as homework. The students should be able to discover on their own that the analysis of n^2 songs are all "isomorphic", and you can easily argue that this is also true of n^2 algorithms. The "isomorphism" of these songs is made precise in the notation we use later (section 4), although I do not present this notation to students.

3. LATER CLASSROOM EXAMPLES

While n^2 songs can be analyzed more quickly than n^2 algorithms, this is even more true of $n \lg n$ and $n^2 \lg n$ songs vs. algorithms. (Throughout this paper, we use "$\lg n$" to stand for $\log_{10} n$ and "$\ln n$" to stand for $\log_e n$.) We are often hard pressed to find easily presented algorithms whose analysis can be done via summations (i.e. not recurrence relations) which are NOT $\bigcirc(n^2)$. The next section gives several examples of songs that are easily analyzed via summations and are either $\bigcirc(n \lg n)$ or $\bigcirc(n^2 \lg n)$. These examples give even more opportunity to discuss the importance of the asymptotics of $\bigcirc()$ notation, and problems that lie therein, e.g., a song that is $\bigcirc(n \lg n)$, but whose $\bigcirc(n)$ term dominates for $n < 10^{23}$. When the students imagine singing *This Old Man*, with 31 syllables constant in all verses, and a single instance of the number "n" (where the number of syllables in n is approximately $\frac{8}{3} \cdot \lg n$), they get a good sense of situations where $\bigcirc()$ notation simplifies matters a bit too much! Including the detail of "how many syllables does it take to sing the number n" is quite similar to considering the space it takes to store the number n, or the time to process it bit by bit. In some cases this is a level of detail best ignored, and in other cases (e.g. radix sort), it is a matter of great importance.

4. DETAILS & EXTENDED ANALYSIS

In this section, we list several examples of songs which can be used either in lecture, or as homework problems. We try to give reasonably exact values of $T(n)$, but these values often use the "average length" of lines or verses, and assume that those values will extend to "invented" verses. In most cases we give only the two highest degree terms of $T(n)$. With many folk songs there are various ways to sing them, and you may calculate a somewhat different value ("your mileage may vary"). The full text of the lyrics used to calculate these values are available at our ftp site:

ftp://MathDeptQuadra/Public/CS/Songs/*

$\bigcirc(n^2)$ Songs:

These songs almost all have the property that the n-th verse includes both a portion specific to that verse, along with n pieces referring to each verse from 1 to n. For example, a common format is for the n-th verse to have the following structure:

Verse $1 = S_1$	R_1	U_1
Verse $2 = S_2$	R_2, R_1	U_2
Verse $3 = S_3$	R_3, R_2, R_1	U_3
	...	
Verse $n = S_n$	$R_n, R_{n-1}, \ldots R_3, R_2, R_1$	U_n

For example, the song Alouette has this pattern with:

S_i = Alouette, gentille Alouette,
Alouette, je te plumerai.
Je te plumerai $\langle X_i \rangle$, je te plumerai $\langle X_i \rangle$,
R_i = Et $\langle X_i \rangle$, et $\langle X_i \rangle$
U_i = Alouette, alouette, Oh —
X_1 = la tête
X_2 = le bec
X_3 = les yeux

X_i = ⟨ two syllables ⟩

If S_i, R_i, and U_i require an average of S, R, and U syllables to sing, then the number of syllables in the first n verses of this song will be:

$$T(n) = (S+U)\cdot n + R\cdot\frac{n(n+1)}{2}$$
$$= \left(S+U+\frac{R}{2}\right)n + \frac{R}{2}\cdot n^2$$

Of course, we don't present this general framework to the students; we want them to do a few songs, then realize that they're all pretty much the same. If we're lucky, they'll extend that realization to analyzing algorithms, many of which also fall into easily analyzed categories.

Examples of songs in this category are listed below, arranged by how large n needs to be for the quadratic term to dominate the linear term.

I Know An Old Lady Who Swallowed a Fly
$S+U=6,\ R=18$ breakpoint = 2

Ich Bin Der Musikant? [Best, p. 88]
$S+U=21,\ R=14$ breakpoint = 4

Old MacDonald Had a Farm
$S+U=37,\ R=22$ breakpoint = 5

There was a Tree Stood in the Ground [Newell, p. 111]
$S+U=36,\ R=11$ breakpoint = 8

There's a Hole in the Bottom of the Sea
$S+U=30,\ R=6$ breakpoint = 11

Alouette [Best, p. 86]
$S+U=40,\ R=7$ breakpoint = 15

B I N G O
This is an interesting example, since the length of a verse (measured in syllables, not in time) *decreases* as the song continues, since each verse sings one fewer letter in the word "Bingo." To generalize this song, we must start with a word containing $n-1$ letters, so that the entire song takes n verses. In this case, we get:
$S+U=22+3n,\ R=-3$ breakpoint = 16
(*John Brown's Baby* is similar, but it is more difficult to decide what "verse n" should be.)

Is das nicht ein Schnitzelbank? [Best, p. 90]
$S+U=47,\ R=6$ breakpoint = 17

She'll Be Coming 'Round the Mountain
$S+U=53,\ R=2$ breakpoint = 53

$\bigcirc(n\lg n)$ Songs:

Songs that contain verses of equal length, except for the existence of the number n in the n-th verse, require $\bigcirc(n\lg n)$ syllables to sing. The detail of the number of syllables required to sing the number n is often viewed as insignificant by students, since it is easy to assume this is $\bigcirc(1)$. As mentioned earlier, I compare this with the amount of storage required to store n, which is constant for many problems, but is $\bigcirc(\lg n)$ if n is effectively unbounded. To convince students that this $\bigcirc(n\lg n)$ factor can sometimes be significant, I have them compare singing "99 bottles of beer on the wall" with singing "1,275,349 bottles of beer on the wall." On the other hand, other songs in this section give clear examples where the asymptotically dominant $\bigcirc(n\lg n)$ factor is less significant than the linear factor for reasonably sized values of n, i.e. where the break-even point between those two terms is so large.

We let $|\#n|$ be the number of syllables required to speak the number n (in English). If we ignore numbers which contain the digit 7 (with 2 syllables) or 0 (resulting in 0 syllables) or the digit 1 when used to make a "teen" number (as with the first "1" in 213,156), then the exact number of syllables required to speak the number i ($i<10^{36}$) is given by:

$$|\#n| = \lceil\log_{10} n\rceil + 2\cdot\lfloor\log_{1000} n\rfloor + 2\cdot\lfloor\log_{1000}(10n)\rfloor$$
$$+\lfloor\log_{1000}(100n)\rfloor + \lfloor\max(0,\log_{1000}(n)-4)\rfloor$$

(= digit symbols + "thousand, million, billion, trillion" + "hundred" + "-ty, as in "thirty" + the extra syllable in "quadrillion" ... "decillion".)

This is a good example where approximate answers are more useful than exact answers. The underestimate from ignoring "7" and the overestimate by ignoring the other exceptions nearly cancel each other. $\lfloor\log_{1000} n\rfloor \cong \frac{1}{3}(\lfloor\lg n\rfloor - 1)$. Using this, and ignoring the final term for $n > 10^{15}$, this formula simplifies to:

$$|\#n| \cong \frac{8}{3}\cdot\lfloor\lg n\rfloor$$

Depending on the class approach, these values can be summed as is, or approximated as:

$$\sum_{i=1}^{n}|\#i| \cong \sum_{i=1}^{n}\frac{8}{3}\cdot\lg i \cong \int_1^n \frac{8}{3}\lg x\,dx + \bigcirc(\lg n)$$
$$= \frac{8\lg e}{3}\cdot(n\ln n - n) + \bigcirc(\lg n)$$

This approximation is used for the values of $T(n)$ reported below.

For the $\bigcirc(n\lg n)$ songs, the number of syllables needed to sing the entire song is $\sum_{i=1}^{n}(V + k|\#i|)$, where V is the number of syllables in a verse other than those used to sing the number i, and k is the number of times i is sung in one verse. Using the approximation above, this simplifies to:

$$T(n) \cong nV + \frac{8k\lg e}{3}\cdot(n\ln n - n)$$

(Again, we do not present this general solution in class, but use or assign one or more specific examples.)

For the songs below, we list the values of V, k, and the "break-point" where the $\bigcirc(n \lg n)$ factor in $T(n)$ is larger than the $\bigcirc(n)$ factor, which happens when $n = exp\left[\frac{3V \ln 10}{8k} - 1\right]$. (However, see the discussion of *The N Days of Christmas* for disclaimers about these break-points.)

Counting Apple-seeds [Newell, p. 109]
$V = 3$ (on average), $k = 1$ breakpoint $= 5$
N Bottles of Beer on the Wall
$V = 26$, $k = 3$ breakpoint $= 655$
N Little Squirrels [Pinsky, p. 9]
$V = 9$, $k = 1$ breakpoint $= 873$
N Little Ducks [Pinsky, p. 10]
$V = 30$, $k = 2$ breakpoint $= 155,134$
N Little Monkeys
$V = 16$, $k = 1$ breakpoint $= 367,880$
N Freckled Frogs
$V = 40$, $k = 2$ breakpoint $= 1 \cdot 10^7$
There Were N in the Bed
$V = 27$, $k = 1$ breakpoint $= 5 \cdot 10^9$
This Old Man [Pinsky, p. 9]
$V = 31$, $k = 1$ breakpoint $= 2 \cdot 10^{11}$
N Men Slept in a Boarding House Bed [Best, p. 73]
$V = 65$, $k = 1$ breakpoint $= 9 \cdot 10^{23}$

$\bigcirc(n^2 \lg n)$ Songs:

When we combine the "verse-building" behavior of the $\bigcirc(n^2)$ songs with the inclusion of sung numbers as in the $\bigcirc(n \lg n)$ songs, the results have $T(n) = \bigcirc(n^2 \lg n)$, and we give two examples of such songs here, using the same approximations as in the previous sub-section.

The N Days of Christmas:

We express the pattern of *The 12 Days of Christmas* using "#i" to represent the ordinal number i, and the format used with the $\bigcirc(n^2)$ songs earlier:

S_i = On the #i day of Christmas, my true love gave to me
$U_i = \emptyset$
R_1 = A partridge in a pear tree
R_2 = Two turtle doves, and
R_3 = Three French hens,
R_4 = Four calling birds,
R_5 = Five gold rings,
R_i = #i + ⟨ four syllables ⟩ $\forall i \geq 6$.

If we let $|S_i|$ and $|R_i|$ be the number of syllables in these parts of the i-th verse, then after the first few verses we have:

$$|S_i| = |\#i| + 12 \cong \frac{8}{3} \cdot \lg i + 12$$
$$\text{and } |R_i| = |\#i| + 4 \cong \frac{8}{3} \cdot \lg i + 4$$

Summing these values we find that the number of syllables in verse n, V_n, is:

$$V_n = \frac{8 \lg e}{3} \cdot (n \ln n - n) + 4n + o(n)$$

Summing these verses, the time to sing *The N Days of Christmas* is:

$$T(n) = \frac{4 \lg e}{3} \cdot n^2 \ln n + 2(1 - \lg e)n^2 + o(n^2)$$
$$\cong 0.58n^2 \ln n + 1.13n^2 + o(n^2)$$

where the n^2 dominates only until $n = 7$.

As hinted at in the previous sub-section, this "break-point" should be interpreted carefully. It is easy to misinterpret this point as the place where the number of syllables devoted to singing numbers begins to dominate the other syllables. This is not the case, and students should notice this with this song, if not earlier. The summation formula for the number syllables includes a negative contribution to the $\bigcirc(n^2)$ term, hence the break-even point between $\bigcirc()$ terms comes much sooner thant the break-even point between "numeric" and "non-numeric" syllables. In addition, the break-point is fairly sensitive to the approximations made earlier, hence this value should only be interpreted as the order of magnitude of the actual break-point.

Green Grow the Rushes O: [Best, p. 32]

This song does not have quite as regular a pattern as *The 12 Days of Christmas*. With a few liberties, however, the pattern can be expressed like that for *The 12 Days of Christmas* with

S_i = I sing you #i, oh. Green grow the rushes, oh. What is your #i, oh?
$U_i = \emptyset$
R_i = ⟨#i twice, plus five syllables.⟩

5. BACKGROUND AND SOURCES

In [Knuth 77] (reprinted in [Knuth 84]), Don Knuth offered a tongue-in-cheek description of the Kolmogorov complexity of songs, asking the question "How much memory is required to store the information needed to sing n verses of this song?" The idea was continued, in the same vein, in [Eisemann 85] and in [Gasarch 92]. As with Knuth, the answer

to our question is usually (but not always) $\bigcirc(n)$. Many of the songs that have interesting answers to Knuth's question (but not all) also have interesting answers to our question. Independently, a student in the author's Algorithms class[1] suggested using songs to teach algorithm analysis, and we are indebted to him for suggesting the topic of this paper. The author has used this idea in subsequent Algorithms classes, and been pleased with the results. In this paper we have not included the text of the songs discussed, hoping that many are familiar to the reader. For teaching purposes it is essential that the full text be available to students, especially for students of other cultures or backgrounds. Full text is also necessary to calculate accurate values of the relevant constants. For many songs we give citations that have the tune and the full text. For all songs the full text is available by anonymous ftp as discussed in section 4. We invite readers to email us additional examples to include in this collection, especially those with interesting complexities or songs from other cultures. (We strongly prefer examples that are widely known among children of at least one region or culture.)

6. CONCLUSIONS

We have found the analysis of songs to be a useful tool in teaching students the first principals of the analysis of algorithms. It provides an interesting break from the normal course work, without leading to an extended diversion. There is easy flexibility in the degree to which one uses this theme, and the number of ideas which are introduced through this analogy. To different degrees this idea can be used in CS1, CS2, or an Algorithms course. This approach seems to have the following pedagogical advantages:

- It provides simple examples of "programs" taking longer with larger inputs, where that increasing time is easily quantifiable;
- Even CS1 students can see the analogy between "How long does a program run?" and "How long does it take to sing a song?";
- In one lecture I can take several "algorithms" all the way from high-level description, to the summation form, to a solution;
- Students quickly realize that the interesting part of most analysis is extracting the summation formula from the "algorithm"—its solution is usually formulaic;
- Many summations common in algorithm analysis arise in these songs, and can be discussed simultaneously (in one lecture if desired) using a "real" example to demonstrate them;
- One can be as carefree, or as precise, with constants as desired, and can easily demonstrate where practical behavior varies considerably depending on the size of these constants (e.g., trivial or significant leading terms);
- Some songs give easy examples of the problems with $\bigcirc()$ notation, e.g., those songs where the leading term is dominated by a lower term for most reasonable values of n.

This approach is not a panacea for teaching analysis. One cannot afford to spend too much time on this diversion. It works best for an early introduction to the topic; e.g., it is not particularly useful for discussing amortized analysis or recurrence relations. (Many songs satisfy $T_n = T_{n-1} + f(n)$, but we introduce these recurrences later when we have sufficient "real" algorithms to illustrate these.) Nevertheless, when used in moderation, songs can be a useful pedagogical tool which offers an interesting change of pace from the regular lecture material.

7. REFERENCES

1. D. Best & B. Best, *Song Fest*, Crown Publishers, New York, 1955.
2. K. Eisemann, Letter: "Further Results on The Complexity of Songs", *Comm. ACM*, 1985, 28(3), 235.
3. W. Gasarch, Sample Abstract, Call for papers for the 1993 STOC conference, 1992.
4. D. Knuth, "The Complexity of Songs", *SIGACT News*, Summer 1977, 17-24.
5. D. Knuth, "The Complexity of Songs", *Comm. ACM*, 1984, 27(4), 344-346.
6. D. Pinsky, *Finger Plays and Action Verses for Children*, Cooperative Extension Service, Iowa State University, 1977.
7. W. W. Newell, *Games and Songs of American Children*, Harper & Brothers, 1883. Reprinted by Dover, 1963, and by Clearfield, 1992.

[1] John Payson, currently at Blue Feather.