A Spelling Correction Program Based on a Noisy Channel Model

Mark D. Kernighan
Kenneth W. Church
William A. Gale

AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, N.J., USA

Abstract

This paper describes a new program, *correct*, which takes words rejected by the Unix® *spell* program, proposes a list of candidate corrections, and sorts them by probability. The probability scores are the novel contribution of this work. Probabilities are based on a noisy channel model. It is assumed that the typist knows what words he or she wants to type but some noise is added on the way to the keyboard (in the form of typos and spelling errors). Using a classic Bayesian argument of the kind that is very popular in the speech recognition literature (Jelinek, 1985), one can often recover the intended correction, *c*, from a typo, *t*, by finding the correction *c* that maximizes $Pr(c)\ Pr(t|c)$. The first factor, $Pr(c)$, is a prior model of word probabilities; the second factor, $Pr(t|c)$, is a model of the noisy channel that accounts for spelling transformations on letter sequences (e.g., insertions, deletions, substitutions and reversals). Both sets of probabilities were trained on data collected from the Associated Press (AP) newswire. This text is ideally suited for this purpose since it contains a large number of typos (about two thousand per month).

1. Introduction

The *correct* program reads a list of misspelled words from the input stream (*stdin*) and prints a set of candidate corrections for each word on the output stream (*stdout*). *Correct* also produces a probability along with each correction (unless there is only one candidate correction). Here is some sample output produced by the Unix® command, ‘‘spell < paper | correct,’’ where *paper* is a text file containing the misspelled words in column 1:

Typo	Corrections
detered	deterred (100%) metered (0%) petered (0%)
laywer	lawyer (100%) layer (0%) lawer (0%)
negotations	negotiations
notcampaigning	???[1]
progession	progression (94%) procession (4%) profession (2%)
ususally	usually
winky	windy (69%) wink (20%) winks (7%) kinky (2%) wonky (1%) pinky (1%) dinky (0%) winy (0%) inky (0%)

2. Proposing Candidate Corrections

The first stage of *correct* finds words that differ from the typo *t* by a single insertion, deletion, substitution or reversal. For example, given the input typo, *acress*, the first stage generates candidate corrections in the table below. Thus, the correct word *actress* could be transformed by the noisy channel into the typo *acress* by replacing the *t* with nothing @ at position 2.[2] This unusually difficult example was selected to illustrate the four transformations; most typo have just a few possible corrections, and there is rarely more than one plausible correction.

Typo	Correction	Transformation			
acress	actress	@	t	2	deletion
acress	cress	a	#	0	insertion
acress	caress	ac	ca	0	reversal
acress	access	r	c	2	substitution
acress	across	e	o	3	substitution
acress	acres	s	#	4	insertion

1. *???* indicates that no correction was found.
2. The symbols @ and # represent nulls in the typo and correction, respectively. The transformations are named from the point of view of the correction, not the typo.

acress	acres	s	# 5	insertion

3. Scoring

Each candidate correction, c, is scored by $Pr(c)\ Pr(t|c)$, and then normalized by the sum of the scores for all proposed candidates. The prior, $Pr(c)$, is estimated by $(freq(c) + 1)/N$, where $freq(c)$ is the number of times that the word c appears in the 1988 AP corpus (N = 44 million words).[3]

The conditional probabilities, $Pr(t|c)$, are computed from four confusion matrices (see appendix): (1) $del[x,y]$, the number of times that the character y was deleted after the character x in the training set, (2), $add[x,y]$, the number of times that y was inserted after x, (3) $sub[x,y]$, the number of times that y (from the correct word) was typed as x, and (4) $rev[x,y]$, the number of times that xy was reversed. Probabilities are estimated from these matrices by dividing by $chars[x,y]$ and $chars[x]$, the number of times that xy and x appeared in the training set, respectively.[4]

$$Pr(t|c) \approx \begin{cases} \dfrac{del[c_{p-1},\ cor_p]}{chars[c_{p-1},\ c_p]}, \text{ if deletion} \\ \dfrac{add[c_{p-1},\ t_p]}{chars[c_{p-1}]}, \text{ if insertion} \\ \dfrac{sub[t_p,\ c_p]}{chars[c_p]}, \text{ if substitution} \\ \dfrac{rev[c_p,\ c_{p+1}]}{chars[c_p,\ c_{p+1}]}, \text{ if reversal} \end{cases}$$

The five matrices are computed with a bootstrapping procedure. Initially assume a uniform distribution over the possible confusions. Then run the program over the training set (1988 AP corpus) to find corrections for the words that *spell* rejects. Use these corrections to update the confusion matrices, and iterate. The matrices are smoothed using the Good-Turing method (Good, 1953).

3. A count of 1 is added in the numerator in order to avoid assigning zero probability to a word. See (Gale and Church, forthcoming) for a discussion of what's wrong with this.

4. The *chars* matrices can be easily replicated, and are therefore omitted from the appendix.

Returning to the *acress* example, the seven proposed transformations are scored by multipling the prior probability (which is proportial to 1 + column 4 in the table below) and the channel probability (column 5) to form a raw score (column 2), which are normalized to produce probabilities (column 1). The final results is: *acres* (45%), *actress* (37%), *across* (18%), *access* (0%), *caress* (0%), *cress* (0%). This example is very hard; in fact, the second choice is probably right, as can be seen from the context: *...was called a ''stellar and versatile* **acress** *whose combination of sass and glamour has defined her....* The program would need a much better prior model in order to handle this case. In the future, a program might be able to take advantage of the fact that *actress* is a considerably more plausible than *acres* as an antecedent for *whose*.

c	**%**	**Raw**	**freq(c)**	**Pr(t|c)**
actress	37%	.16	1343	55./470,000
cress	0%	˜0	0	46./32,000,000
caress	0%	˜0	4	.95/580,000
access	0%	˜0	2280	.98/4,700,000
across	18%	.077	8436	93./10,000,000
acres	21%	.092	2879	417./13,000,000
acres	23%	.098	2879	205./6,000,000

4. Evaluation

Many typos such *absorbant* have just one candidate correction, but others such as *adjusted* are more difficult and have multiple corrections. The table below shows examples of typos with less than ten candidate corrections.

#	**Typo**	**Corrections**
0	admininistration	
1	absorbant	absorbent
2	adusted	adjusted dusted
3	ambitios	ambitious ambitions ambition
4	compatability	compatibility compactability comparability computability
5	afte	after fate aft ate ante
6	dialy	daily diary dials dial dimly dilly
7	poice	police price voice poise pice ponce poire
8	piots	pilots pivots riots plots pits pots pints pious
9	spash	splash smash slash spasm stash swash sash pash spas

Most typos have relatively few candidate corrections. The table below shows the number of typos[5] broken out by the number of corrections in seven month-long samples of the AP newswire. In March, for example, there were 720 typos with 0 corrections, 1120 typos with 1 correction, 269 with 2 corrections, etc. The final column shows that there is a general trend for fewer choices, though the 0-choice case is special. (The system was trained on the AP wire from 2/88 - 2/89; the results below were computed from AP wire during 3/89 - 9/89).

#	**March**	**April**	**May**	**June**	**July**	**Aug**	**Sept**	**Total**
0	720	604	542	606	492	465	508	3937
1	1120	997	1037	1007	958	944	930	6993
2	269	224	209	223	199	224	214	1562
3	109	92	89	101	79	87	82	639
4	58	57	62	45	43	59	43	367
5	54	41	20	26	28	24	28	221
6	35	22	19	19	22	17	23	157
7	20	11	13	7	11	15	17	94
8	19	14	14	5	7	7	16	82
9	15	11	6	11	10	8	16	77
10+	154	97	79	75	53	77	78	613
Total	2573	2170	2090	2125	1902	1927	1955	14,742

We decided to look at the 2-candidate case in more detail in order to test how often the top scoring candidate agreed with a panel of three judges. The judges were given 564 triples and a few concordance lines:

absurb absorb absurd

financial community . *E* *S* ‘‘ It is absurb and probably obscene for any person so engaged to und

The first word of the triple was a *spell* reject; the other two were the candidates (in alphabetical order). The judges were given a 5-way forced choice. They could circle any one of the three words, if they thought that was what the author had intended. Alternatively, if they thought that the author had intended something else, they could write down ‘‘other’’. Finally, if they weren’t sure, they could write ‘‘?’’. The distribution of responses is shown in the following table.

	Judge 1	**Judge 2**	**Judge 3**
choice 0	99	124	93
choice 1	188	176	167
choice 2	175	159	151
other	28	26	30
?	74	79	123
total	564	564	564

The results show that *spell* is rejecting too many words, since choice 0 (spell error) is selected about 20% of the time. In these cases, *correct* was given a non-problem to correct:

acquirees acquirers acquires

be acquirers , as they have been , than acquirees . *E* *S* If the industrials had attracted bids th

Since we were mostly concerned with evaluating the scoring function, we didn’t want to be distracted with errors in *spell* and other problems that are beyond the scope of this paper. Therefore, we decided to consider only those cases where at least two judges circled one of the two candidates, and they agreed with each other. This left 332 triples.

The following table shows that *correct* agrees with the majority of the judges in 87% of the 332 cases of interest. In order to help calibrate this result, three inferior methods are also evaluated. The *no-prior* method ignores the prior probability. The *no-channel* method ignores the channel probability. Finally, the *neither* method ignores both probabilities and selects the first candidate in all cases. As the following table shows, *correct* is significantly better than the three inferior alternatives. Both the channel and the prior probabilities provide a significant contribution, and the combination of the two is signicantly better than either in isolation. The second half of the table evaluates the judges against one another and shows that they significantly out-perform *correct*, indicating that there is plenty of room for further improvement.[6] All three judges found the task more difficult and more time consuming than they had expected. Each judge spent about half a

5. For the purposes of this experiment, a typo is a lowercase word rejected by the Unix® spell program.

6. Judges were not scored on a triple if they selected ‘‘?,’’ ‘‘other’’ or ‘‘spell error’’; They were only evaluated on a triple if they selected one of the two proposed candidates. A triple was graded ‘‘correct’’ if it agreed with the majority opinion, and ‘‘incorrect’’ if it did not.

day grading the 564 triples.

Method	Discrimination	σ	%
correct	288/332	± 6.2	87%
no-prior	267/332	± 7.2	80%
prior-only	254/332	± 7.8	77%
alphabetic	172/332	± 9.1	52%
Judge 1	326/328	± 1.4	99%
Judge 2	317/321	± 2.0	99%
Judge 3	292/302	± 3.2	97%

We were also interested in testing whether the score predicted accuracy. Figure 1 (at the end of this paper) shows that this is indeed so. The horizontal axis shows the score from *correct* averaged over a group of typos. The vertical axis shows (a smooth of) the fraction of this group that agreed with the majority opinion of the judges. The straight line indicates perfection. As you can see, the line corresponding to *correct* (labeled with 1's) follows the perfection line fairly well. The other two lines show that the no-prior method (labeled with *p*'s) and the no-channel method (labeled with *c*'s) are not nearly as good at predicting their own accuracy.

5. Conclusions

There have been a number of spelling correction programs in the past such as Kucera (1988) that generated a list of candidates by looking for insertions, deletions, substitutions and reversals, much as we have been doing here. Our contribution is the emphasis on scoring. Doug McIlroy, the author of the Unix® *spell* program, intentionally focused on the spelling detection problem, and argued that spelling correction was a bad idea so long as the corrector couldn't separate the plausible candidates from the implausible ones. He felt that it was probably more distracting than helpful to bury the user under a long list of mostly implausible candidates. In this work, we have attempted to show that it is possible to sort the candidates by a likelihood function that agrees well enough with human judges to be helpful.

In future work, we would hope to extend the prior model to take advantage of context. We noticed that the human judges were extremely reluctant to cast a vote given only the information available to the program, and that they were much more comfortable when they could see a concordance line or two. Perhaps our program could take advantage of these contextual cues by adopting very simple language modeling techniques such as trigrams, that have proven effective for speech recognition applications (Jelinek, 1985). Hopefully more interesting language models would improve performance even more.

References

Gale, W., Church, K., (forthcoming), ''What's Wrong with Adding One?''

Good, I.J., (1953), ''The population frequencies of species and the estimation of population parameters,'' *Biometrika*, v. 40, pp. 237-264.

Jelinek, F. (1985) ''Self-organized Language Modeling for Speech Recognition,'' IBM Report.

Kucera, H., (1988), ''Automated Word Substitution Using Numerical Rankings of Structural Disparity Between Misspelled Words & Candidate Substitution Words,'' Patent Number: 4,783,758.

McIlroy, M., (1982), ''Development of a Spelling List,'' *IEEE Transactions on Communications*, Vol. COM-30, No. 1.

6. Appendix: Confusion Matrices

del[X, Y] = Deletion of Y after X

Y (Deleted Letter)

X	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	7	58	21	3	5	18	8	61	0	4	43	5	53	0	9	0	98	28	53	62	1	0	0	2	0
b	2	2	1	0	22	0	0	0	183	0	0	26	0	0	2	0	0	6	17	0	6	1	0	0	0	0
c	37	0	70	0	63	0	0	24	320	0	9	17	0	0	33	0	0	46	6	54	17	0	0	0	1	0
d	12	0	7	25	45	0	10	0	62	1	1	8	4	3	3	0	0	11	1	0	3	2	0	0	6	0
e	80	1	50	74	89	3	1	1	6	0	0	32	9	76	19	9	1	237	223	34	8	2	1	7	1	0
f	4	0	0	0	13	46	0	0	79	0	0	12	0	0	4	0	0	11	0	8	1	0	0	0	1	0
g	25	0	0	2	83	1	37	25	39	0	0	3	0	29	4	0	0	52	7	1	22	0	0	0	1	0
h	15	12	1	3	20	0	0	25	24	0	0	7	1	9	22	0	0	15	1	26	0	0	1	0	1	0
i	26	1	60	26	23	1	9	0	1	0	0	38	14	82	41	7	0	16	71	64	1	1	0	0	1	7
j	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0
k	4	0	0	1	15	1	8	1	5	0	1	3	0	17	0	0	0	1	5	0	0	0	1	0	0	0
l	24	0	1	6	48	0	0	0	217	0	0	211	2	0	29	0	0	2	12	7	3	2	0	0	11	0
m	15	10	0	0	33	0	0	1	42	0	0	0	180	7	7	31	0	0	9	0	4	0	0	0	0	0
n	21	0	42	71	68	1	160	0	191	0	0	0	17	144	21	0	0	0	127	87	43	1	1	0	2	0
o	11	4	3	6	8	0	5	0	4	1	0	13	9	70	26	20	0	98	20	13	47	2	5	0	1	0
p	25	0	0	0	22	0	0	12	15	0	0	28	1	0	30	93	0	58	1	18	2	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0
r	63	4	12	19	188	0	11	5	132	0	3	33	7	157	21	2	0	277	103	68	0	10	1	0	27	0
s	16	0	27	0	74	1	0	18	231	0	0	2	1	0	30	30	0	4	265	124	21	0	0	0	1	0
t	24	1	2	0	76	1	7	49	427	0	0	31	3	3	11	1	0	203	5	137	14	0	4	0	2	0
u	26	6	9	10	15	0	1	0	28	0	0	39	2	111	1	0	0	129	31	66	0	0	0	0	1	0
v	9	0	0	0	58	0	0	0	31	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	1	0
w	40	0	0	1	11	1	0	11	15	0	0	1	0	2	2	0	0	2	24	0	0	0	0	0	0	0
x	1	0	17	0	3	0	0	1	0	0	0	0	0	0	0	6	0	0	0	5	0	0	0	0	1	0
y	2	1	34	0	2	0	1	0	1	0	0	1	2	1	1	1	0	0	17	1	0	0	1	0	0	0
z	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
@	20	14	41	31	20	20	7	6	20	3	6	22	16	5	5	17	0	28	26	6	2	1	24	0	0	2

add[X, Y] = Insertion of Y after X

Y (Inserted Letter)

X	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	15	1	14	7	10	0	1	1	33	1	4	31	2	39	12	4	3	28	134	7	28	0	1	1	4	1
b	3	11	0	0	7	0	1	0	50	0	0	15	0	1	1	0	0	5	16	0	0	3	0	0	0	0
c	19	0	54	1	13	0	0	18	50	0	3	1	1	1	7	1	0	7	25	7	8	4	0	1	0	0
d	18	0	3	17	14	2	0	0	9	0	0	6	1	9	13	0	0	6	119	0	0	0	0	0	5	0
e	39	2	8	76	147	2	0	1	4	0	3	4	6	27	5	1	0	83	417	6	4	1	10	2	8	0
f	1	0	0	0	2	27	1	0	12	0	0	10	0	0	0	0	0	5	23	0	1	0	0	0	1	0
g	8	0	0	0	5	1	5	12	8	0	0	2	0	1	1	0	1	5	69	2	3	0	1	0	0	0
h	4	1	0	1	24	0	10	18	17	2	0	1	0	1	4	0	0	16	24	22	1	0	5	0	3	0
i	10	3	13	13	25	0	1	1	69	2	1	17	11	33	27	1	0	9	30	29	11	0	0	1	0	1
j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
k	2	4	0	1	9	0	0	1	1	0	1	1	0	0	2	1	0	0	95	0	1	0	0	0	4	0
l	3	1	0	1	38	0	0	0	79	0	2	128	1	0	7	0	0	0	97	7	3	1	0	0	2	0
m	11	1	1	0	17	0	0	1	6	0	1	0	102	44	7	2	0	0	47	1	2	0	1	0	0	0
n	15	5	7	13	52	4	17	0	34	0	1	1	26	99	12	0	0	2	156	53	1	1	0	0	1	0
o	14	1	1	3	7	2	1	0	28	1	0	6	3	13	64	30	0	16	59	4	19	1	0	0	1	1
p	23	0	1	1	10	0	0	20	3	0	0	2	0	0	26	70	0	29	52	9	1	1	1	0	0	0
q	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
r	15	2	1	0	89	1	1	2	64	0	0	5	9	7	10	0	0	132	273	29	7	0	1	0	10	0
s	13	1	7	20	41	0	1	50	101	0	2	2	10	7	3	1	0	1	205	49	7	0	1	0	7	0
t	39	0	0	3	65	1	10	24	59	1	0	6	3	1	23	1	0	54	264	183	11	0	5	0	6	0
u	15	0	3	0	9	0	0	1	24	1	1	3	3	9	1	3	0	49	19	27	26	0	0	2	3	0
v	0	2	0	0	36	0	0	0	10	0	0	1	0	1	0	1	0	0	0	0	1	5	1	0	0	0
w	0	0	0	1	10	0	0	1	1	0	1	1	0	2	0	0	1	1	8	0	2	0	4	0	0	0
x	0	0	18	0	1	0	0	6	1	0	0	0	1	0	3	0	0	0	2	0	0	0	0	1	0	0
y	5	1	2	0	3	0	0	0	2	0	0	1	1	6	0	0	0	1	33	1	13	0	1	0	2	0
z	2	0	0	0	5	1	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	4
@	46	8	9	8	26	11	14	3	5	1	17	5	6	2	2	10	0	6	23	2	11	1	2	1	1	2

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X \ Y (correct)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

rev[X, Y] = Reversal of XY

X \ Y	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	2	1	1	0	0	0	19	0	1	14	4	25	10	3	0	27	3	5	31	0	0	0	0	0
b	0	0	0	0	2	0	0	0	0	0	0	1	1	0	2	0	0	0	2	0	0	0	0	0	0	0
c	0	0	0	0	1	0	0	1	85	0	0	15	0	0	13	0	0	0	3	0	7	0	0	0	0	0
d	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	1	0	0	2	0	0	0	0	0
e	1	0	4	5	0	0	0	0	60	0	0	21	6	16	11	2	0	29	5	0	85	0	0	0	2	0
f	0	0	0	0	0	0	0	0	12	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
g	4	0	0	0	2	0	0	0	0	0	0	1	0	15	0	0	0	3	0	0	3	0	0	0	0	0
h	12	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0
i	15	8	31	3	66	1	3	0	0	0	0	9	0	5	11	0	1	13	42	35	0	6	0	0	0	3
j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
k	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l	11	0	0	12	20	0	1	0	4	0	0	0	0	0	1	3	0	0	1	1	3	9	0	0	7	0
m	9	0	0	0	20	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	4	0	0	0	0	0
n	15	0	6	2	12	0	8	0	1	0	0	0	3	0	0	0	0	0	6	4	0	0	0	0	0	0
o	5	0	2	0	4	0	0	0	5	0	0	1	0	5	0	1	0	11	1	1	0	0	7	1	0	0
p	17	0	0	0	4	0	0	1	0	0	0	0	0	0	1	0	0	5	3	6	0	0	0	0	0	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	12	0	0	0	24	0	3	0	14	0	2	2	0	7	30	1	0	0	0	2	10	0	0	0	2	0
s	4	0	0	0	9	0	0	5	15	0	0	5	2	0	1	22	0	0	0	1	3	0	0	0	16	0
t	4	0	3	0	4	0	0	21	49	0	0	4	0	0	3	0	0	5	0	0	11	0	2	0	0	0
u	22	0	5	1	1	0	2	0	2	0	0	2	1	0	20	2	0	11	11	2	0	0	0	0	0	0
v	0	0	0	0	1	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
w	0	0	0	0	0	0	0	4	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	8	0
x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
y	0	1	2	0	0	0	1	0	0	0	0	3	0	0	0	2	0	1	10	0	0	0	0	0	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0