Principe de construction de l'arbre binaire

Etat de départ : N paires, $2^{n-1} \leq N < 2^n$
arbre de degré N tiré de la liste des paires :
racine + N feuilles, aucun nœud groupement.

La séquence suivante réduit de un le degré de l'arbre.
La répéter pour créer $N - 2$ nœuds.

- Choisir deux fils (gauche, droit) de la racine et les en détacher
- Créer un nouveau nœud pour les regrouper : augmente de un la profondeur de toutes leurs feuilles.
- Attacher le nouveau nœud à la racine.

Etat final : arbre binaire = racine + N feuilles + N-2 nœuds

Rappel des objectifs

Huffman : arbre de codage optimal avec borne Lmax = N-1

Package-merge : un arbre de codage optimal sous contrainte :

Lmax borné : 8*sizeof(. . . type entier . . .)

Hauteur limitée : raisonnement

Départ : un arbre binaire équilibré et complet.

Toutes les feuilles sont à la même profondeur Lmax.

Si $N = 2^{Lmax}$: codage des symboles en taille fixe
(toutes les feuilles de profondeur Lmax sont utilisées).

Si $N > 2^{L}max$: codage impossible (pas assez de feuilles).

Suppression de feuilles

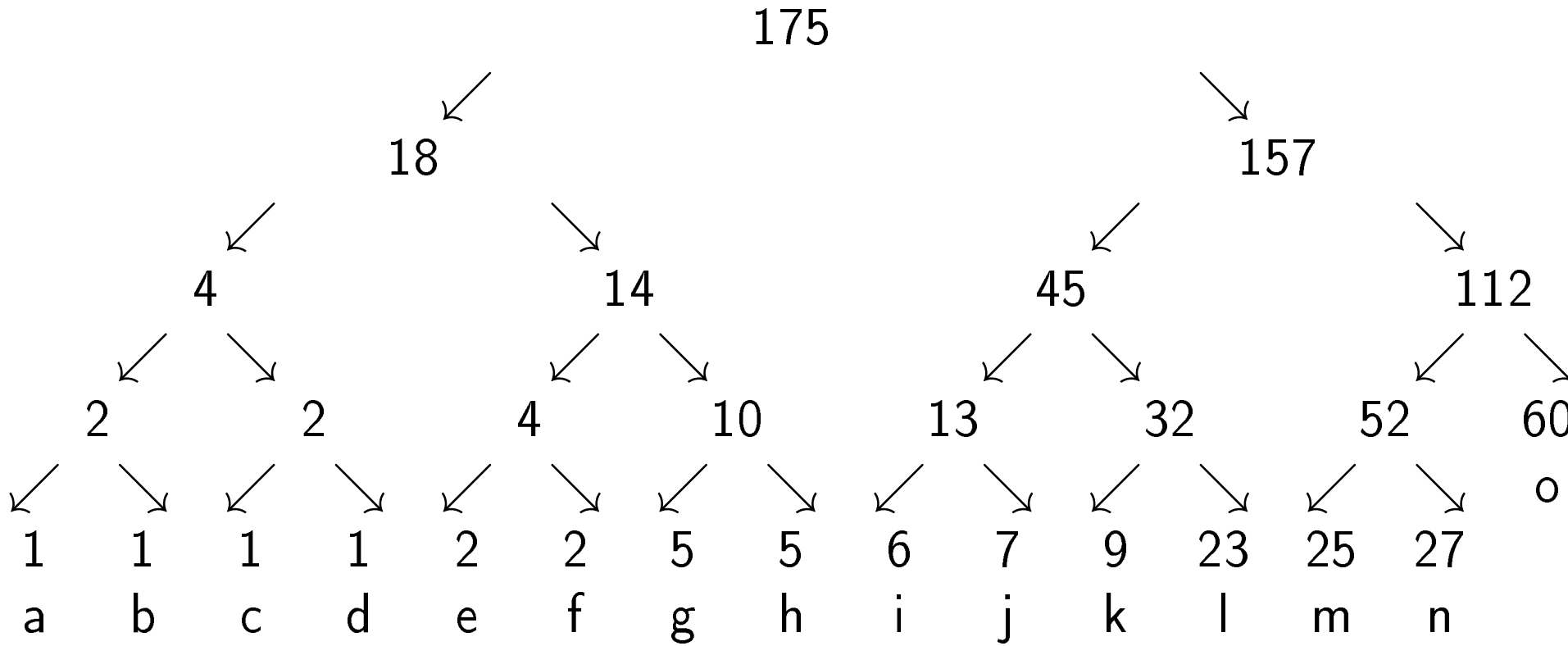
Un codage en taille variable (et gain de taille) devient possible lorsque $N < 2^{Lmax}$:

- $\exists$ un sous-arbre contenant un unique symbole S :
- supprimer le sous-arbre
- remplacer sa racine par la feuille-symbole S
- effet : la longueur du code de S diminue.

Quel(s) symbole(s) remonter ?

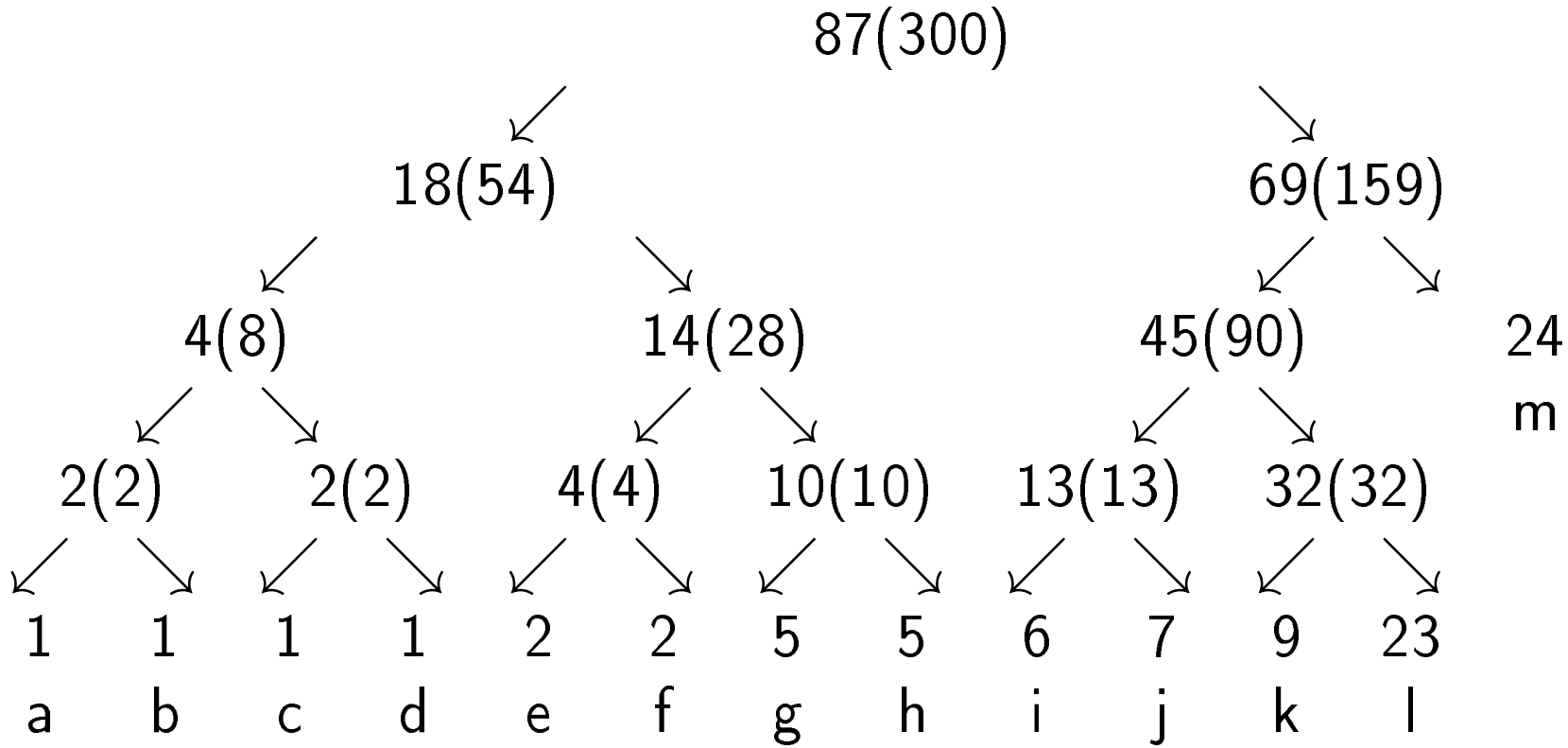
Cas simple : une seule feuille libre $\rightarrow$ le + fréquent

Exemple 15 symboles / une feuille libre



Il manque un symbole pour saturer l'arbre
$\rightarrow$ le symbole o remonte d'un niveau.

Exemple 13 symboles / 16 cases non optimal



Idée simpl(ist)e : pas optimale
remonter d'abord au maxi le + fréquent
puis remonter au max le suivant + fréquent, etc

Critique de 13 symboles / 16 cases non optimal

Règle d'ordre des groupements et des feuilles :

- poids croissant
- à poids égal ordre croissant de première feuille

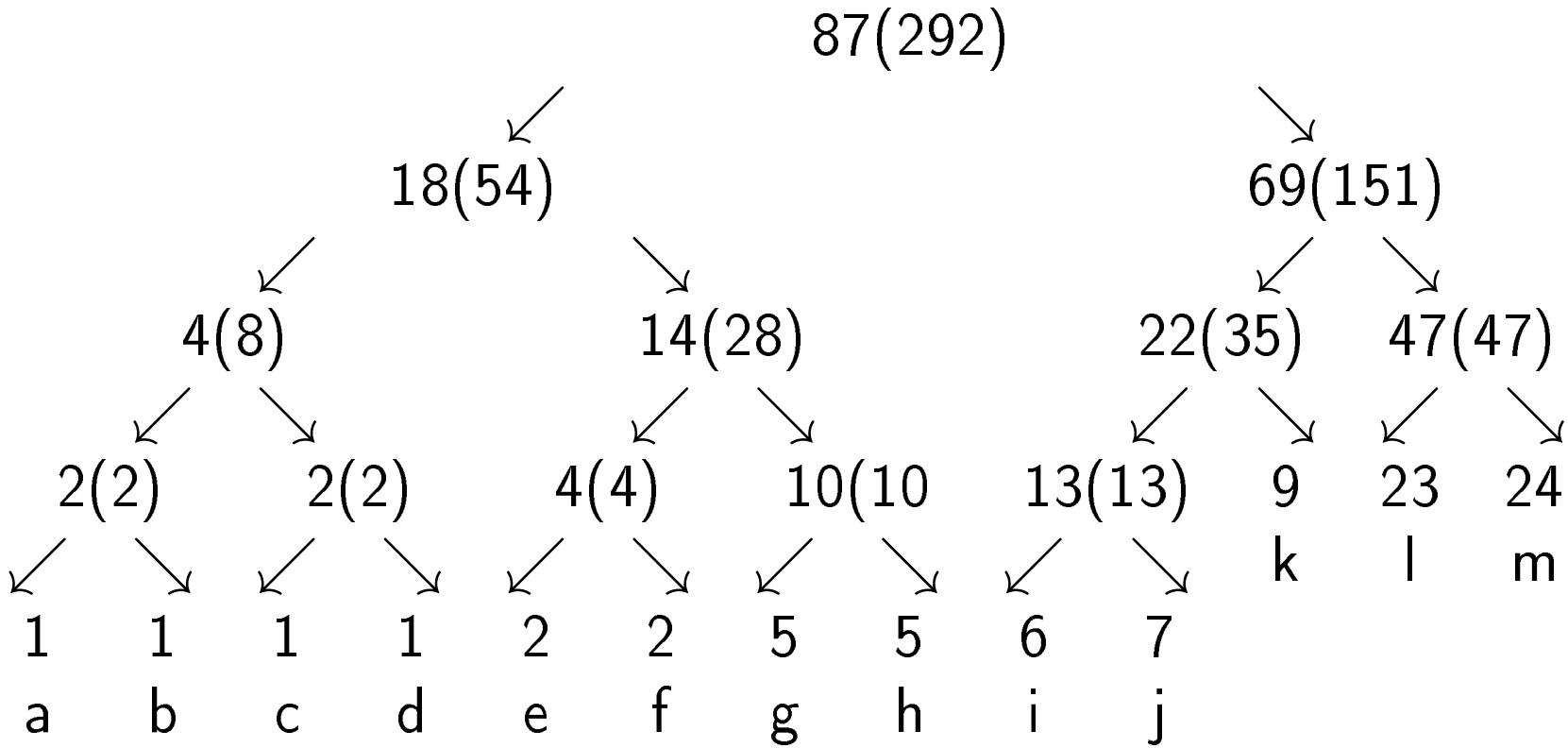
Une passe pourrait-elle remettre en cause les paires de feuilles issues de la première étape (groupement de symboles) ?

Quel groupements pourraient venir décaler les mises en paires en s'intercalant entre les feuilles ?

- ij ou gh entre k et l
- ef devant g
- ab et cd devant e

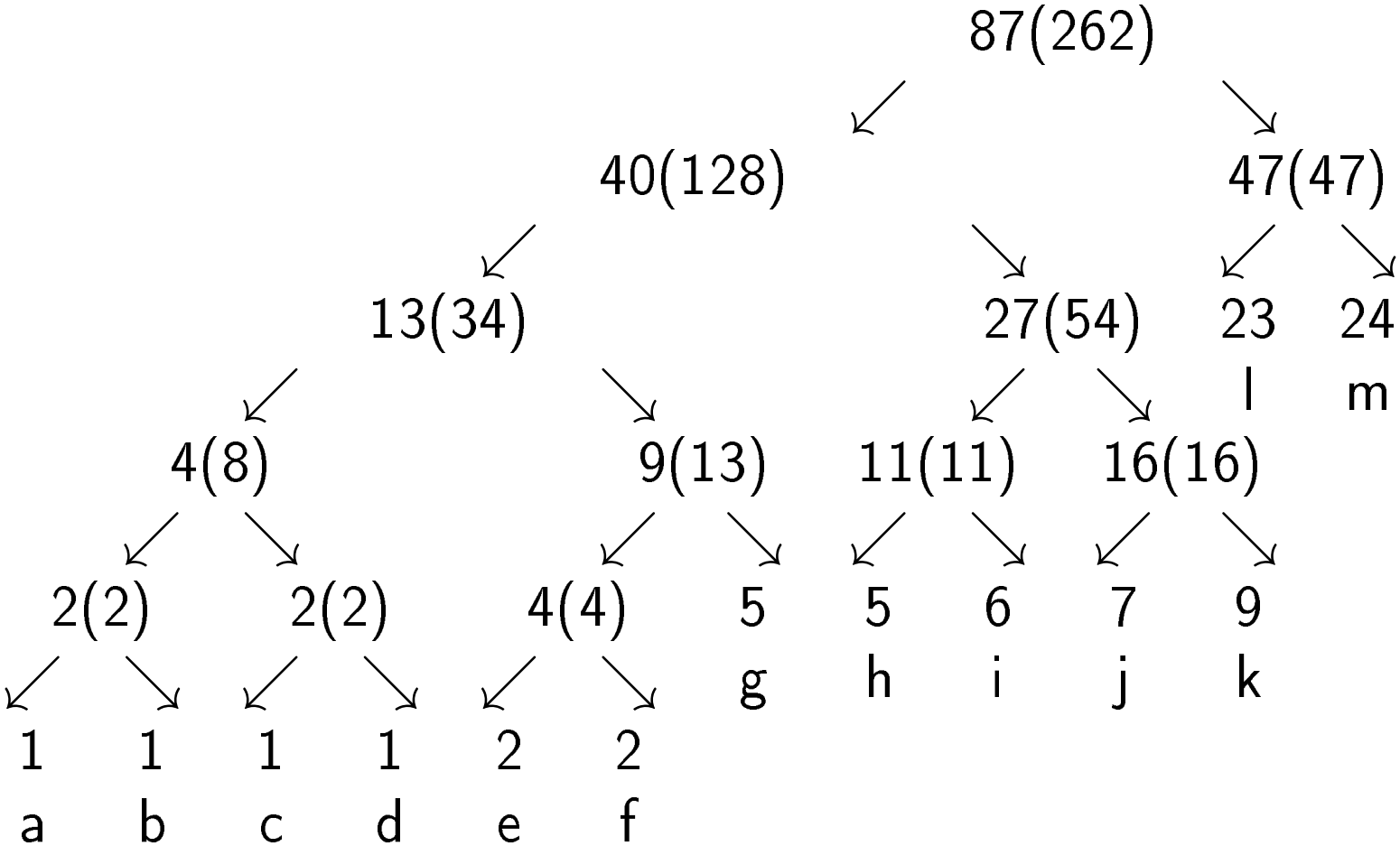
Le décalage introduit par l'insertion de ij permet de générer des groupements alternatifs moins coûteux lors des étapes suivantes.

Exemple 13 symb. / 16 optimal



Longueurs : a à j $\rightarrow$ 4, k,l,m $\rightarrow$ 3

Exemple 13 symb. / 32 cases



Longueurs de codes : a à f : $\rightarrow$ 5, g à k $\rightarrow$ 4, l,m $\rightarrow$ 2

Principe de package-merge

Schéma de base : grouper par 2^1, puis par 2^2, ... par 2^i ... pour former l'arbre équilibré complet de profondeur Lmax.

Chaque pas de calcul est une étape de groupement qui incrémente la profondeur maximale des feuilles symboles.

L'étape initiale ne groupe que des feuilles symboles.

Les groupements issus d'une étape s'ajoutent aux candidats à appairer. L'arbre initialement équilibré se déforme progressivement quand un groupement issu d'une étape de calcul peut s'intercaler entre les paires du schéma de base.

Les pas de calcul ajoutent des groupements permis par l'augmentation de la profondeur courante : ne conserver que les N-2 moins coûteux et ignorer les autres.

Algorithme package-merge

Conditions initiales : Ajouts = { }

Ordre : poids croissant, ordre croissant symbole le moins fréquent.

Répéter pour l=2 à Lmax :

- Liste = { }
- Pour f parcourant la liste des feuilles
  - Insérer_en_triant (Liste,f)
- Pour f parcourant Ajouts
  - Insérer_en_triant (Liste,f) : merge
  - Retirer (Ajouts, f)
- Ajouts = appairer (Liste) : package

Parcourir les N-2 premières paires de Ajouts

Longueur code de feuille = 1 + nombre de présences dans groupements

Exemple package-merge détaillé

1	1	1	1	2	2	5	5	6	7	9	23	24	
a	b	c	d	e	f	g	h	i	j	k	l	m	Liste

2	2	4	10	13	32	
ab	cd	ef	gh	ij	kl	Ajouts (package)

1	1	1	1	2	2	2	2	4	5	
a	b	c	d	↑	↑	e	f	↑	g	Liste (merge)
				ab	cd			ef		

5	6	7	9	10	13	23	24	32
h	i	j	k	↑	↑	l	m	↑
				gh	ij			kl

2	2	4	4	9	11	16	23	47	
ab	cd	abcd	ef	efg	hi	jk	ghij	lm	Ajouts

1	1	1	1	2	2	2	2	4	4		
a	b	c	d	↑	↑	e	f	↑	↑		Liste
				ab	cd			abcd	ef		
5	5	6	7	9	9	11	16	23	23	24	47
g	h	i	j	↑	k	↑	↑	↑	l	m	↑
				efg		hi	jk	ghij			lm

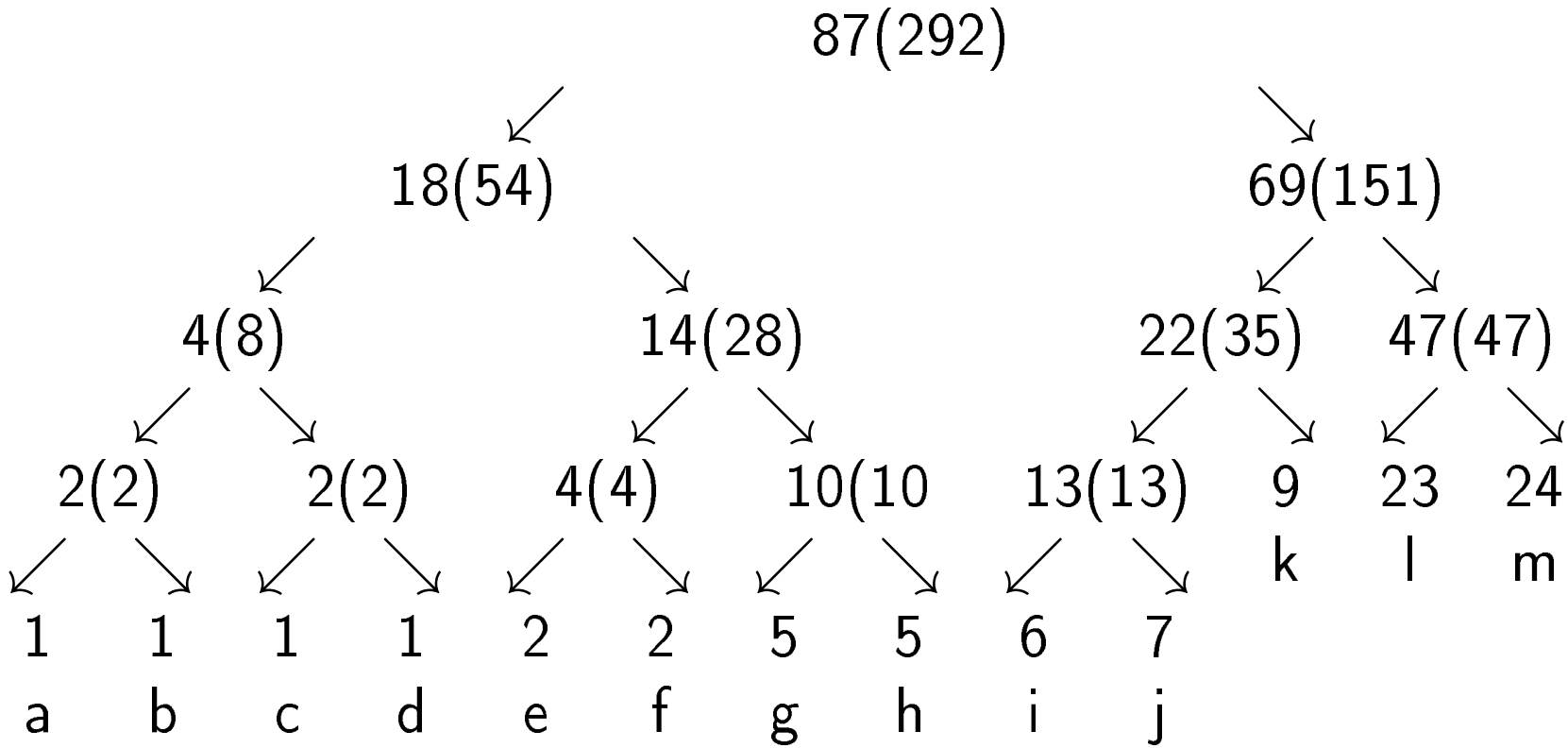
2	2	4	4	8	10	13	18	27	46	71
ab	cd	abcd	ef	abcdef	gh	ij	efgk	hijk	ghijl	mlm
1	2	3	4	5	6	7	8	9	10	11

Exactement N-2 groupements

a,b,c,d,e,f,g,h,i,j : cités chacun 3 fois parmi groupes → L=4

k,l,m : cités chacun 2 fois parmi groupes → L=3

Exemple 13 symb. / 16 optimal



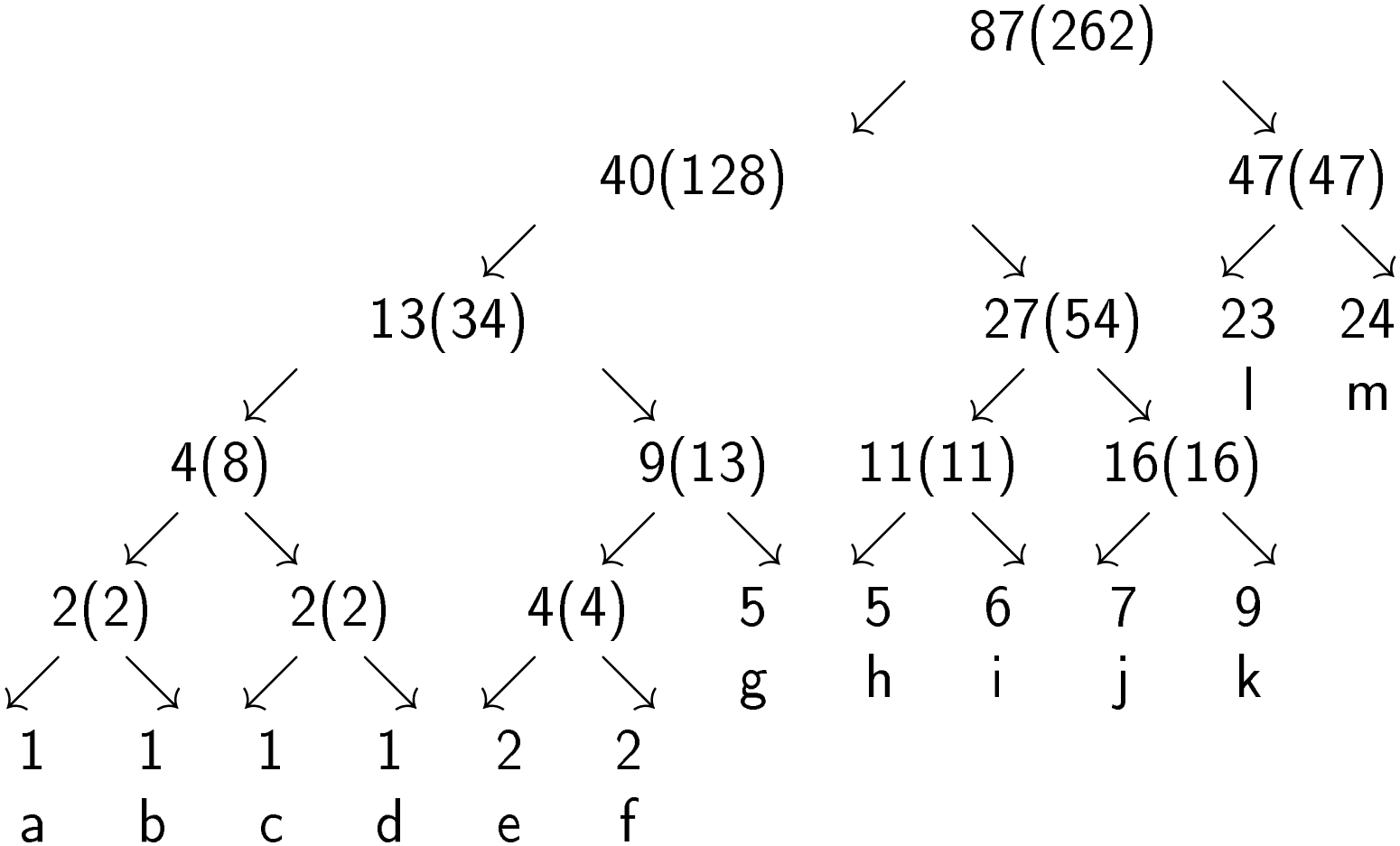
Longueurs : a à j $\rightarrow$ 4, k,l,m $\rightarrow$ 3

1	1	1	1	2	2	2	2	4	4	5	5
a	b	c	d	↑	↑	e	f	↑	↑	g	h
				ab	cd			abcd	ef		
6	7	8	9	10	13	18	23	24	27	46	71
i	j	↑	k	↑	↑	↑	l	m	↑	↑	↑
		abcdef		gh	ij	efgk			hijk	ghijl	mlm

2	2	4	4	8	10	13	17	23	41	51
ab	cd	abcd	ef	abcdef	gh	ij	abcdefk	ghij	efgkl	hijkm
1	2	3	4	5	6	7	8	9	10	11

a,b,c,d,e,f cités 4 fois chacun dans les groupes : L=5
g,h,i,j,k cités 3 fois chacun dans les groupes : L=4
l,m cités une fois chacun : L=2

Exemple 13 symb. / 32 cases



Longueurs de codes : a à f : $\rightarrow$ 5, g à k $\rightarrow$ 4, l,m $\rightarrow$ 2

1	1	1	1	2	2	2	2	4	4	5	5
a	b	c	d	↑	↑	e	f	↑	↑	g	h
				ab	cd			abcd	ef		

6	7	8	9	10	13	17	23	23	24	41	51
i	j	↑	k	↑	↑	↑	↑	l	m	↑	↑
		abcdef		gh	ij	abcdefk	ghij			efgkl	hijkm

2	2	4	4	8	10	13
ab	cd	abcd	ef	abcdef	gh	ij
1	2	3	4	5	6	7

17	23	40	47
abcdefk	ghij	abcdefkghij	lm
8	9	10	11

a,b,c,d : cités chacun 5 fois $\rightarrow$ L=6
e,f : cités chacun 4 fois $\rightarrow$ L=5
g,h,i,j : cités chacun 4 fois $\rightarrow$ L=4
k : cité 2 fois $\rightarrow$ L=3
l,m : cités chacun une fois $\rightarrow$ L=2

L'arbre et le codage sont identiques au résultat de l'algorithme de Huffman.

Un autre exemple d'arbre de Huffman