アルゴリズムとデータ構造II

講義8：

アルゴリズム設計手法：
欲張りアルゴリズム

https://elms.u-aizu.ac.jp

欲張りアルゴリズム

私 最適化問題のアルゴリズムは通常、
一連のステップを経て、各ステップで一連の選択肢があります。

私 A 欲張りアルゴリズム 常にその選択をします
見える 現時点で最高。つまり、この選択がグローバルに最適なソリューションにつながることを期待して、ローカルに最適な選択を行います。

欲張りアルゴリズムの例

私 欲張りアルゴリズムの例

私 プリムのアルゴリズムとクラスカルのアルゴリズム

私 最小全域木問題

最短の単一ソース問題に対するダイクストラのアルゴリズム

パスの問題。

私 今、私たちは貪欲の別のアプリケーションを紹介します

アルゴリズム、データ圧縮の設計（ヒューマン）コード。

ファイル圧縮

私 26文字のA、B、...、Zが割り当てられていると仮定します

次の5ビットバイナリコード：（A、00001）（1）、（B、00010）（2）、（C、00011）（3）、...、（Z、11010）（26）。

私 ABRACADABRAが与えられた場合、5ビットのバイナリを使用します

表現、それは次のようにエンコードされます：

アブラカダブラ

00001000101001000010001100001

00100000010001010010000001

私 このメッセージをデコードするには、一度に5ビットずつ読み取ります。

上記の割り当てに従って変換します。

私 シーケンスでは、Dが1回表示され、Aが5回表示されます。

両方ともバイナリコードで5ビットが割り当てられています。

シーケンスを短くする

私 シーケンスを短くするアイデアは、を割り当てることです
最も頻繁に出現する文字の最短ビット文字列。

私 「ABRACADABRA」では、Aが5回出現し、Bが出現します。
2回、Cが1回、Dが1回、Rが2回表示されます。

シーケンスを短くします（続き）

私 （A、0）、（B、1）、（R、01）、
（C、10）、（D、11）、およびABRACADABRAは010101001101010として
エンコードされます

私 これは、以前の55ビットと比較して15ビットを使用します。
ただし、このコードは空白に依存して文字を区切ります。ブランクがないと、文字列010101001101010はRRRARBRRAまたは他のいくつかの文字列としてデコードできます。

シーケンスを短くする（続き2）

私 を使用せずに文字列をコンパクトにエンコードする方法

私 区切り文字 文字のエンコードがどのコードのプレフィックスでもない場合
他の文字の場合、区切り文字は必要ありません。

私

私 0、(A、0)、および(B、1)、(R、01)、(C、10)、(D、11）では、
AはRの接頭辞であり、BのコードはCとDの接頭辞です。

シーケンスを短くする（続き3）

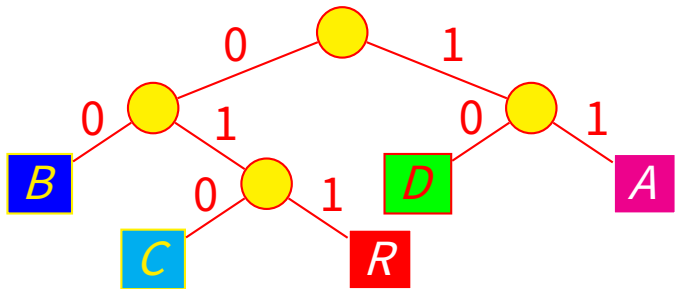
私 M枚の葉を持つ二分木を使用してエンコードできます
M個の異なる文字を含むメッセージ（1つのリーフが1つの文字を保持します）。

私 各文字のコードはパスによって決定されます
ルートからその文字を保持するリーフまで、0は「左」に移動し、1は「右」に移動します。

シーケンスを短くする（続き4）

私 たとえば、次の二分木はエンコードします *A* なので

11、 *B* なので 00、 *C* なので 010、 *D* なので 10、および *R* なので 011。

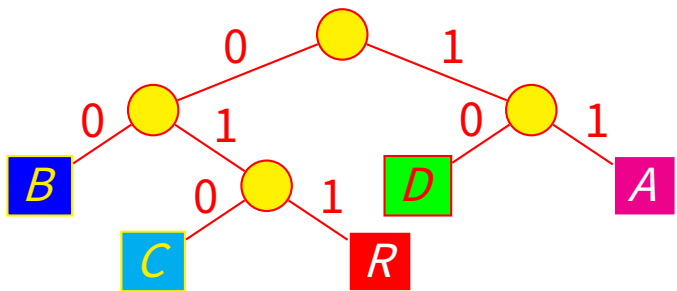


ABRACADABRA としてエンコードされます
11000111101011101100001111 25ビット

シーケンスを短くする（続き5）

私 二分木表現は、

文字コードは別のプレフィックスであるため、文字列はツリーから一意にデコードできます。

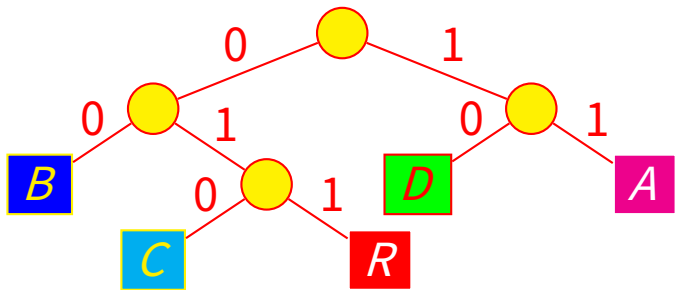


シーケンスを短くする（続き6）

私 ルートから始めて、ツリーを下に進みます

メッセージのビットに：リーフが検出されるたびに、そのリーフで文字を出力し、

ルートから再起動します。

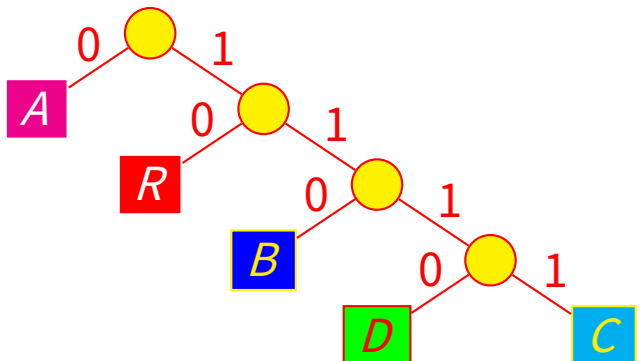


11000111101011101100001111

シーケンスを短くする（続き7）

私 以下は、エンコード用の別のバイナリツリーです

ABRACADABRA。



01101001111011100110100 23ビット

ハフマン符号化

私 質問：どの二分木が使用するのに最適か
（指定された文字列に対して最短のバイナリシーケンスを提供します）。

私 D.ヒューマンはエレガントな方法を与えました（ヒューマンエンコーディング）任意の文字列に対して最小長のビット文字列につながる二分木を計算します。

ハフマン符号化

私 ハフマン符号化は、貪欲なアプローチを使用して、

エンコード用のバイナリツリー。

私 しましょう S特定の文字に表示される文字のセットである

ストリング。

ハフマン符号化の手順

私 ステップ1、の各文字の頻度を数える S に

指定された文字列。検討するS二分木の根のセットとして、各ツリーには1つのノードがあります。

私 ステップ2、最小頻度の2つの根を見つける

2つのルートのルートとして新しいノードを生成します。新しいルートには、2人の息子の頻度の合計の頻度があります。から2人の息子を削除します

S新しいルートをに追加します S。 手順2を、次のようになるまで繰り返します。

$/S/=1.1$。

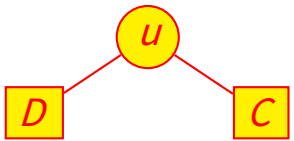
ハフマン符号化の手順

私 与えられた文字列 アブラカダブラ。S = {A、B、C、D、R}、

w（A）=5、 w（B）=2、 w（C）=1、 w（D）=1、 w（R）=2.2。

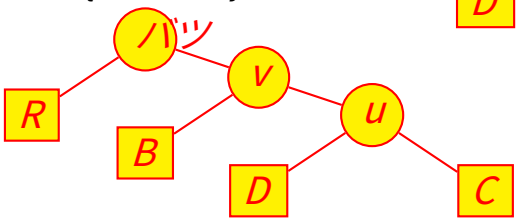
5 2 1 1 2

S = {A、 B、 C 、D、 R}

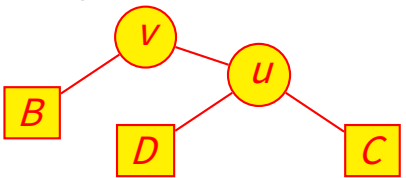


5 4 2

S = {A、 v、 R}



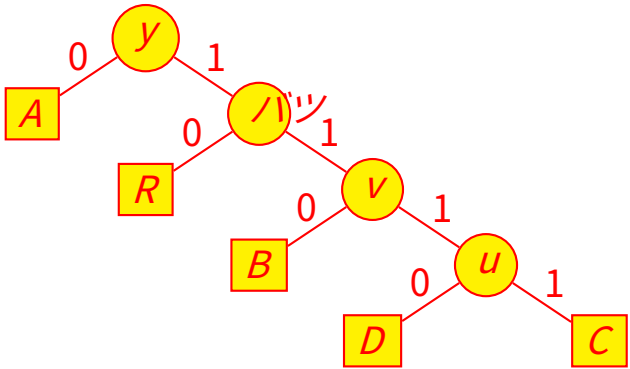
S = {5A、2.2B、2 u、R}



ハフマン符号化の手順

5 6

$S = \{A、\ バツ\}$



11

$S = \{y\}$

私 の文字にバイナリコードを割り当てます Sなので：

（（A、 0）、 （（B、 110）、 （（C、 1111）、 （（D、 1110）、および （（R、 10）。

0110100111101110011010023ビット

もう一つの例

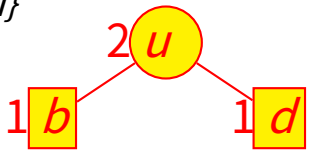
私 与えられた文字列「*aaabccdef*」。

$S = \{a、b、c、d、e、f\}$、w（a）=3、 w（b）=1、 w（c）=2、 w（d）=1、 w（e）=1、および w（f）=1.欲張りアプローチを使用して、次のスライドに示すように二分木を取得します。

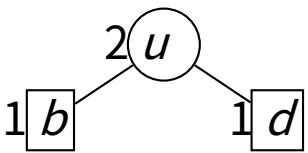
別の例（続き）

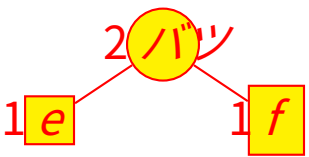
$S = \{a、b、c、d、e、f\}$

$S = \{a、c、e、f、u\}$



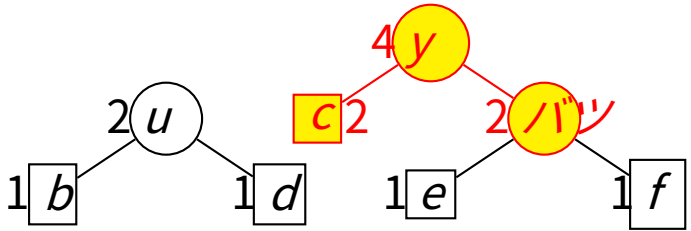
$S = \{a、c、u、x\}$



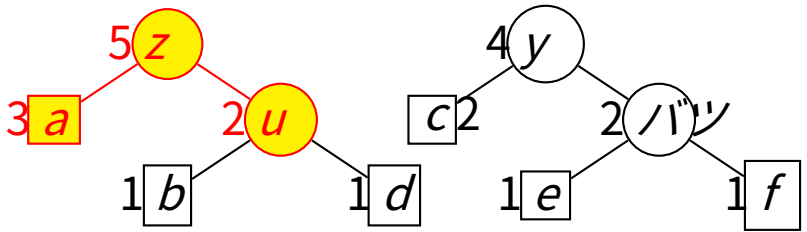


別の例（続き　　2）

$S = \{a、u、y\}$

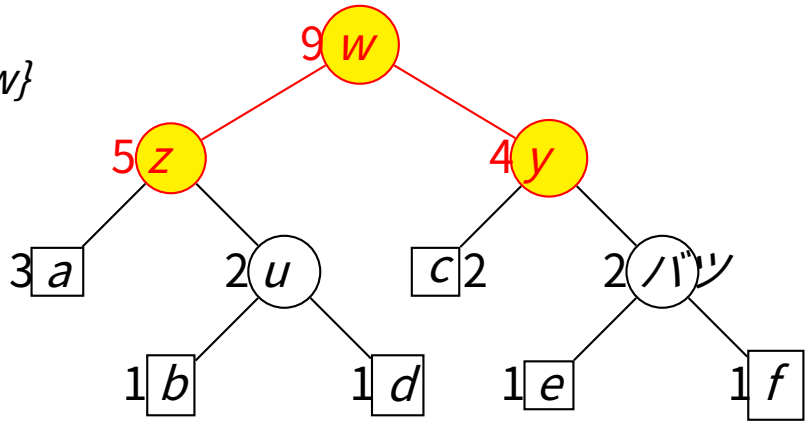


$S = \{y、z\}$

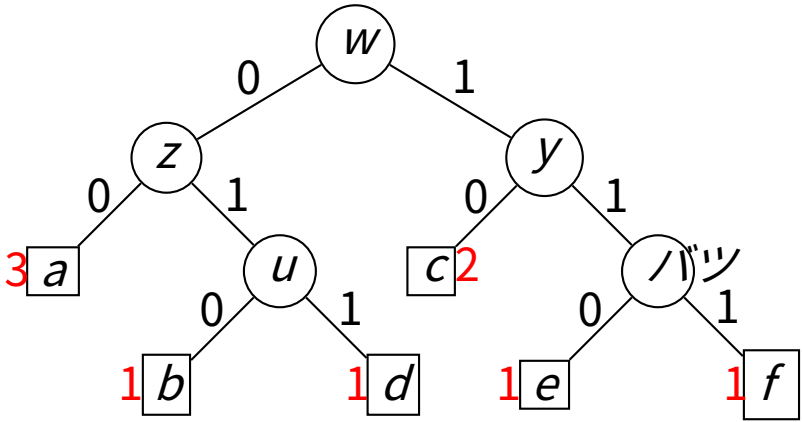


別の例（続き3）

$S = \{w\}$



別の　　例（続き4）



割り当てます　Sの文字へのバイナリコードは次のとおりです。

（a、00）、（b、010）、（c、10）、（d、011）、（e、110）、（f、111）。

別の例（続き5）

私 エンコーディング用の二分木を見つける際に、私たちは繰り返し
セットから最小周波数を持つ2つの根を見つける S 見つかった根をから削除します S、 次に、挿入する新しいルートを作成します S。

私 明らかに、ヒープのデータ構造は
それらの操作。

想起：ダウンヒープ

```
void downheap（int k、int n）{
    int j、u、v;
    u =ヒープ[k]; v =インデックス[k];
    while（k <= n / 2）{
        j = k + k;
        if（j <n && heap [j]> heap [j + 1]）j ++; if
          （u <= heap [j]）break;
        heap [k] = heap [j]; index [k] = index [j]; k
        = j; }
    heap [k] = u; index [k] = v;}
```