自然言語処理 108－18
(1995. 7. 21)

日本語会話文の言語解析実験

田代敏久　森元 逞
ATR 音声翻訳通信研究所
〒619-02 京都府相楽郡精華町光台2丁目2番地
0774-95-1301
tashiro@itl.atr.co.jp, morimoto@itl.atr.co.jp

あらまし

頑健かつ高精度で、処理効率が良い構文・意味解析機構を目指して、多種多様な言語知識を有効に利用でき、外部モジュールと容易にリンクできる構文解析ツールキットの開発を進めている。本報告では、構文解析ツールキットを用いた句構造解析、依存構造解析、及び格構造解析の三つの解析実験の概要及び実験結果を報告する。さらに、依存構造解析や格構造解析結果を定量的に評価するために、一般的な木構造を対象とする新しい評価尺度を提案する。

キーワード:　構文・意味解析、句構造、依存構造、格構造、評価尺度

Parsing Experiments of Spoken-Style Japanese Sentences

Toshihisa Tashiro, Tsuyoshi Morimoto
ATR Interpreting Telecommunications Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, JAPAN
0774-95-1301
tashiro@itl.atr.co.jp,　morimoto@itl.atr.co.jp

Abstract

To obtain a robust, accurate and efficient parser, we are developing a parsing toolkit, which can be linked with external modules easily and can make a good use of various knowledge sources. In this paper, we report the outlines and the results of three parsing experiments: a phrase structure analysis, a dependency structure analysis, and a case structure analysis. Additionally, to evaluate the results of the dependency structure and the case structure analysis quantitatively, we propose a new evaluation method of which target is a general tree structure.

key words:　parser, phrase structure, dependency structure, case structure, evaluation method

1 はじめに

頑健かつ高精度で、処理効率が良い構文・意味解析機構を目指して、多種多様な言語知識を有効に利用でき、外部モジュールと容易にリンクできる構文解析ツールキットの開発を進めている。我々は既に、文献[1]において、構文解析ツールキットの概要及び簡単な句構造解析実験の結果について報告した。本報告では、句構造解析、依存構造解析、格構造解析の三つのモジュールの内容をより詳細に説明し、各モジュールを使った解析実験の結果を報告する。

2 構文解析ツールキットの概要

文献[1]で述べたように、従来の構文意味解析機構の研究は、

- 計算機構のメカニズムの精密さを過度に追求するあまり、現実のデータに対する実験的な検討が不十分である。
- 多様な言語現象の一部分を説明できるに過ぎない文法理論に拘泥するあまり、自然言語を機械的に処理するために本当に有効な知識の検討が不十分である。
- システム開発を過度に優先するあまり、処理メカニズムや使用する言語知識がアドホックなものとなり、システマチックな改良や他のシステムとの客観的な比較が困難である。

等の問題を抱えていることが多い。

そこで我々は、構文解析ツールキットを構築するにあたり、

- 計算メカニズムの形式的な明確さを守りながら、現実のデータに対して動作可能な処理機構を開発する。
- 言語知識を宣言的・形式的に記述する枠組は守りながら、現実のデータの処理に必要な言語知識を開発する。

ことを常に重視している。図1に構文解析ツールキットのイメージを、図2にツールキットを構成するモジュールの概要を示す。

3 句構造解析モジュール

句構造解析モジュールとは、形態素列を入力にとり、句構造規則に従い、構成要素(constituent)をノードとする木構造(句構造)を作成する機構である。図3に句構造解析モジュールの入出力例を示す。

我々は句構造解析を行なう計算メカニズムとして純粋にボトムアップな動作を行なうチャートパーザを、句構造規則として純粋な文脈自由文法を用いている。

(モジュール)	(データ)	(外部へのインターフェース)
	↓	
[形態素解析]		
↑↓		
<形態素調整>	形態素列	⇔ 外部モジュールへ(から)
↑↓		
[句構造解析]	↓	
↑↓		
<構文木調整>	句構造	⇔ 外部モジュールへ(から)
↑↓		
[依存構造解析]	↓	
↑↓	依存構造	⇔ 外部モジュールへ(から)
[格構造解析]	↓	
↑↓	格構造	⇔ 外部モジュールへ(から)
[意味解析]	↓	
	意味構造	⇔ 外部モジュールへ(から)

※<形態素調整> 及び <構文木調整> モジュールは必要に応じて利用する。

図 1: 構文解析キットのイメージ

形態素解析モジュール

基本機能	文字列から形態素列への変換
データ構造変換知識	辞書(形態素辞書)
曖昧性解消知識	単語の N-gram、品詞(ラベル)の N-gram、接続テーブル、最長一致等のヒューリスティック等

形態素調整モジュール

基本機能	形態素列のデータ書き換え
データ書き換え知識	形態素列書き換え規則
曖昧性解消知識	単語の N-gram、品詞(ラベル)の N-gram、接続テーブル、最長一致等のヒューリスティック等

句構造解析モジュール

基本機能	形態素列から句構造への変換
データ構造変換知識	句構造規則(文脈自由文法)
曖昧性解消知識	確率文法、規則の連鎖統計情報、語や句の共起関係、枝分かれに関するヒューリスティック等

構文木調整モジュール

基本機能	句構造のデータ書き換え
データ書き換え知識	句構造書き換え規則
曖昧性解消知識	確率文法、規則の連鎖統計情報、語や句の共起関係、枝分かれ等に関するヒューリスティック等

依存構造解析モジュール

基本機能	句構造から依存構造への変換
データ構造変換知識	依存構造規則(注釈付き句構造規則)
曖昧性解消知識	規則の適用確率、語や句の共起関係、統語的な制約、枝分かれ等に関するヒューリスティック等

格構造解析モジュール

基本機能	依存構造(アークラベルなしグラフ)から格構造(アークラベル付きグラフ)への変換
データ構造変換知識	格情報辞書規則
曖昧性解消知識	意味素性、規則の適用確率、語や句の共起関係、シソーラス、ドメインに依存するヒューリスティック等

意味解析モジュール

基本機能	ラベル付きグラフの書き換え
データ構造変換知識	グラフ書き換え規則
曖昧性解消知識	規則の適用確率、メタ規則等

図 2: 構文解析ツールキットの各モジュールの概要

```
入力:
  ((ニューワシントンホテル <固有名詞>)(で <助動詞>)
   (ございま <補助動詞>)(す <語尾>)(。 <句点>))

出力:
  (<文>
    (<節>
      (<動詞句>
        (<動詞>
          (<名詞句>
            (<固有名詞>  ニューワシントンホテル))
          (<助動詞>  で))
        (<補助動詞>
          (<補助動詞語幹>  ございま)
          (<語尾>  す))))
    (<句点>  。))
```

図 3: 句構造解析モジュールの入出力例

通常のチャート解析においては、無駄な弧の提案を避けるために、ボトムアップとトップダウンの弧の提案機構を併用することが多い。しかし、我々は頑健な解析を行なうために、fitted parse の手法 [2] を利用するために、トップダウンの予測は行なわないことにした。

また、通常の句構造ベースの解析においては、文脈自由文法規則に注釈を加えた拡張文脈自由文法を利用することが多い。一般に、規則の注釈は、

- 規則が適用される環境を制限する。
- 規則が適用された時に作成する依存構造等のより意味的な構造を指定する。

という二つの目的で利用されている。しかし、

- 我々が対象とする日本語会話文には、規則の注釈として記述できるような統語的な制約があまりない。
- 統語構造の作成と意味的な構造の作成を分離して行なうほうが、処理効率及びシステムの保守性の点で好都合である。

という二つの理由により、純粋な文脈自由文法を用いることにした。図 4 に句構造規則の例を示す。

純粋な文脈自由文法を用いることによる曖昧性の増大は、拡張文脈自由文法の注釈のような制約ではなく、選好を利用して解消する [3]。曖昧性解消のための知識には様々なものがあるが、今回の実験では、北 [4] により提案された句構造規則のバイグラムを用いることにした。図 5 に文法規則のバイグラムの例を示す。

現在、約 240 の句構造規則と約 55000 の語彙規則 (形態素辞書) を開発している。また、ATR の音声言語データベース [5] の約 10000 文に対して構文木が付与されており、句構造規則のバイグラムの抽出や解析結果の評価に利用できるようになっている。

```
( <名詞句> <--> ( <サ変名詞> ) )
( <後置詞句> <--> ( <名詞句> <格助詞> ) )
( <動詞> <--> ( <本動詞> <語尾> ) )
```

図 4: 句構造規則の例

4 依存構造解析モジュール

依存構造解析モジュールとは、句構造 (構文木) を入力とし、構成要素 (constituent) の統語的な主従関係を判定し、"語"[1]をノードとするアークラベルを持たない有向グラフ[2]に変換するモジュールである。図 6に依存構造解析モジュールの入出力例を示す。

我々は依存構造解析を、句構造規則に付与されたタイプに固有の手続きを呼び出すことにより実現した。現在実装している手続き (タイプ) は以下の 5 つである。

- TYPE-0:
  句構造規則の右辺の要素の数が一つしかない場合は、単に右辺の情報を伝播する。
- TYPE-1:
  右辺の要素の数が二つで、右側の要素が統語上の head となる場合、左側の要素を右側の要素の子 (娘) 要素とし、右側の要素の情報を伝播する。日本語の句構造規則のほとんどはこのタイプである。
- TYPE-2:
  右辺の要素の数が二つで、左側の要素が統語上の head となる場合、右側の要素を左側の要素の子 (娘) 要素とし、左側の要素の情報を伝播する。日本語の句構造規則にはこのような例はめったに出現しない。
- TYPE-3:
  右辺の要素の数が二つで、二つの要素の間の親子関係は存在しないが、意味的な中心は左側の要素である場合、二つの要素を一つのノードに併合し、左側の要素の情報を伝播する。日本語の句構造規則では、動詞語幹と語尾の規則等が相当する。
- TYPE-4:
  右辺の要素の数が二つで、二つの要素の間に親子関係が存在しないが、意味的な中心は右側の語である場合、二つの要素を一つのノードに併合し、右側の要素の情報を伝播する。日本語の句構造規則では、接頭語と名詞の規則等が相当する。

[1] 通常は単語と考えてよい。動詞等の活用する語は、複数の形態素で一つの単語を構成すると考える。

[2] 殆どの場合、木構造で十分表現できる。

```
((<格助詞> <--> (まで)) (<格助詞> <--> (に))  -1.674)
((<格助詞> <--> (まで)) (<後置詞句> <--> (<名詞句> <格助詞>))  -.30167)
((<格助詞> <--> (まで)) (<副詞句> <--> (<節> <格助詞>))  -4.5643)
```

図 5: 句構造規則のバイグラムの例

```
入力:
    (<文>
      (<節>
        (<動詞句>
          (<動詞>
            (<名詞句>
              (<固有名詞>  ニューワシントンホテル))
            (<助動詞>  で))
          (<補助動詞>
            (<補助動詞語幹>  ございま)
            (<語尾>  す))))
      (<句点>  。))

出力:
  (。):(<句点>)
    └─(ございま す):(<補助動詞語幹> <語尾>)
      └─(で):(<助動詞>)
        └─(ニューワシントンホテル):(<固有名詞>)
```

図 6: 依存構造解析モジュールの入出力例

```
入力:
  (。):(<句点>)
    └─(ございま す):(<補助動詞語幹> <語尾>)
      └─(で):(<助動詞>)
        └─(ニューワシントンホテル):(<固有名詞>)

出力:
 [ [語義見出し  。]
   [カテゴリ  <句点>]
   [任意  [ [語義見出し  *ございます*]
            [カテゴリ  <補助動詞語幹>]
            [任意  [ [語義見出し  *です*]
                     [カテゴリ  <助動詞>]
                     [ＯＢＪＥ  *未定義*]
                     [ＩＤＥＮ
                       [ [語義見出し
                          *ニューワシントンホテル*]
                         [カテゴリ
                           <固有名詞>] ] ] ] ] ] ] ]
```

図 8: 格構造解析モジュールの入出力例

```
( ( <名詞句> <--> ( <サ変名詞> )  ) TYPE-0)
( ( <後置詞句> <--> ( <名詞句> <格助詞> )  ) TYPE-1)
( ( <動詞> <--> ( <本動詞> <語尾> )  ) TYPE-3)
```

図 7: 依存構造規則の例

```
[
借りる
  [AGEN  : [[品詞: 格助詞] [表層語形 :(が)] [意味素: [HUMAN +]]]]
  [OBJE  : [[品詞: 格助詞] [表層語形 :(を)] [意味素: [CONC +]]]]
]
```

図 9: 格解析辞書の例

図7 に依存構造規則の例を示す。現在、約 250 の依存構造規則 (タイプ付き句構造規則) [3] を開発している。また、ATR の音声言語データベースの約 6000 文に対して依存構造木が付与されており、語と語の共起関係の抽出や解析結果の評価に利用できるようになっている。

5 格構造解析モジュール

格構造解析モジュールとは、依存構造 (アークラベルを持たない有向グラフ) を入力とし、格解析辞書を利用して語の主従関係の役割を決定し、格構造 (アークラベルを持つ有向グラフ) に変換するモジュールである。

通常、格構造解析を行なうためには、図9に示すような格フレームを記述した辞書を利用することが多い。我々も同様の手法を採用する予定であるが、現時点では辞書が整備途中であるため、図 10に示すような語 の三項関係を代理の格構造解析規則として用いることにした。なお、この三項関係はあらかじめ人手により格構造解析されたコーパスから自動的に抽出することができる。

[3] 同じ句構造規則にも複数のタイプが付与される場合があるため、句構造規則よりも若干多くなっている。

現時点では抽出された規則をそのまま利用しているが、図 11に示すように規則を一般化することにより、格フレームを記述した辞書と同等の生成能力を持たすことも可能である。また、この三項関係の出現頻度は、格構造解析時や前段階の解析時に生じた曖昧性の解消のために利用することができる。

現在、約 10000 の格構造解析規則 (三項関係) を用意している。また、ATR の音声言語データベースの約 6000 文に対して格構造木が付与されており、規則の抽出や解析結果の評価に利用できるようになっている。

6 解析結果の評価手法

形態素解析や句構造解析の結果を定量的かつ機械的に評価する手法は既に提案されている [6, 7]。しかし、これらの評価法は、表層の語順が解析結果に保持されていることを前提としており、より深い (意味的な) 構造である依存構造や格構造の評価にそのまま利用することは難しい。そのため、従来の依存構造や格構造の評価は、解析結果が正解と完全に一致するか否かという単純な尺度 (文正解率) で行なわれることが多かった。

語1	語1と語2の関係	語2	語2と語3の関係	語3
借りる	OBJE	を	nil	レンタカー

図 10: 格構造解析規則の例

語1	語1と語2の関係	語2	語2と語3の関係	語3の意味素
借りる	OBJE	を	nil	CONC

図 11: 一般化された格構造解析規則の例

しかし、依存構造解析や格構造解析の技術を着実に進歩させるには、よりきめの細かい評価方法を用いて、技術の進歩を客観的な数値としてすことが必要だと思われる。また、実際の自然言語処理システムを構築する上でも、入力を完全に正確に解析できなくても済む場合も多い。そこで、本稿では、依存構造解析や格構造解析の結果に部分的な評価値を与える手法を提案する。

一般に、依存構造や格構造は木構造で表現することができる[4]。そこで、本稿では、依存構造や格構造の評価に、2つの木構造(正解構造とシステム出力構造)に共通する最大部分構造のノードの数を利用することにした。

具体例で説明する。図12は、"I saw a man with a telescope" という文を句構造解析した結果を、Blackの手法[7]により評価した例である。この例では、システムは "with a telescope" の係り先を誤っており、ブラケット単位の整合性を評価すると再現率と適合率は共に5/6という結果となる。

図13は、同じ文を依存構造解析まで行なった結果を、我々の手法で評価した例である。この例では、二つの木構造の最大部分構造は、

```
  saw
 /   \
I    man
      |
      a
```

となり、再現率と適合率は共に4/7となる。

また、句構造解析の評価法は、英語テキストの解析結果を評価するために定められたものなので、正解構造中の形態素と出力構造中の形態素は必ず一対一に対応していることが前提となっている。そのため、日本語のように単語境界が明確でない言語を文字列から解析したり、文字認識や音声認識の結果を解析する場合のように、正解構造中の形態素と出力構造中の形態素が異なる可能性があることを想定していない。しかし、我々の手法は二つの木構造に特別な条件を科さな

[4]様々な言語現象をより的確に記述するためにはグラフ構造が必要であるが、ほとんどの場合木構造で近似することが可能である。

```
正解:
(I ((saw (a man)) (with (a telescope))))

システム:
(I (saw ((a man) (with (a telescope)))))

再現率  5/6 : 適合率  5/6 : 交差数  1
```

図 12: 句構造(ブラケット構造)の評価法

いので、図14のような評価を行なうことも可能である。

付録Aに、任意の2つの木構造を評価する手続きを示す。この手続きでは、入力の木構造がアークラベルを持つ木構造であることを前提としており、本稿でいうところの依存構造のようなアークラベルを持たない木構造は、すべてのアークが同じラベルを持っていると考えて対応する。また、この手続きは、あるノードが同じラベルのアークを複数持っている場合には、必ずしも真の最大共通部分構造のノードの数を求めるわけではない。同じ句が一つの語に複数係っている場合のような特殊なケースでは、真の最大共通部分構造のノードの数よりも小さい値を出力する可能性がある。しかし、このようなケースは実際の自然言語を対象とする限り極めて稀であると思われる。

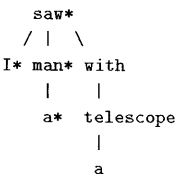
7 実験結果

現在開発を進めている構文解析ツールキットを用いて行なった句構造解析、依存構造解析、及び格構造解析の三つの解析実験結果を報告する。

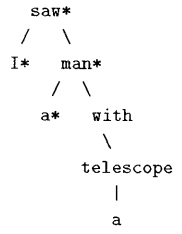
実験はATRの音声言語データベースに含まれている64会話(2352文)を対象に行なった。なお、この64会話は、文法規則の開発や統計情報の抽出時に利用した対話とは別のものである(つまり、今回の実験はすべてオープンテストである)。図15に今回の実験対象の例文を示す。

まず、あらかじめ正しく形態素解析された2352文を

正解:



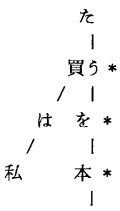
システム:



再現率 = 適合率 = 4/7

図 13: 一般の解析木の評価法

正解:



システム:

再現率 3/7 : 適合率 3/4

図 14: 形態素が異なる場合の解析木の評価法

```
(
    (<感動詞>
        (<副詞句>
            (<副詞>  大変) )
        (<感動詞>
            (<感動詞>  申し訳ございません)
            (<読点>  、) ) )
    (<名詞句>
        (<人名>  鈴木)
        (<接尾辞>  様) )
    (<句点>  。)
)
```

図 16: 不適格文の部分木による表現

再現率 (= 適合率)	文正解率
96.2%	84.6%

表 2: 依存構造解析の実験結果

入力とし、句構造解析を行ない、第一候補の解を Black の方法で評価した結果を表 1に示す。なお、試験対象中には、図 16で示すような統語的に不適格な文や、適格な文でも曖昧性の爆発によるメモリの限界等により一つの木構造として解析できない文も含まれている。2352 文中、一つの木構造として解析できた文は 2049 文 (87.1%) である。

本来は、複数の部分木を意味的な情報を用いて併合 (再構成) する必要があるが、現段階では部分木併合規則は整備途中であるため、依存構造解析及び格構造解析実験は句構造解析で一つの木構造として解析できた文のみを対象とすることにした。表 2に依存構造解析モジュールの解の第一候補を、本稿で提案した一般的な木構造評価手法で評価した結果を示す。

依存構造解析モジュールは、句構造解析で一つの木構造として解析した結果をすべて解析する (何らかの解を出力する) ことができた。この依存構造の出力を格構造解析した結果を表 3 に示す。格解析モジュールも依存構造の出力をすべて解析解析する (何らかの解を出力する) ことができた。結局、最終的な解析成功率 (文正解率) は、42.5%(999/2353) となった。

我々はこの実験結果を、今後行なう計算機構や文法の改良作業の際のベースラインとして考えている。つまり、今回の実験で使ったような単純な計算機構や文法でも、これだけの解析精度はある以上、たとえどんなに高度な計算機構や文法を用いても、今回の実験結果より悪くなるのであればあまり意味がないのではないか、というのが我々の見解である。

はい、シェラトンリゾートでございます。
御用、承ります。
鈴木と申しますが、予約をお願いしたいのですが。
はい、鈴木様。
いつ御到着で、何日間御滞在の予定でしょうか。
九月一日から三日間お願いしたいのですが。
承知いたしました。
では、何名様で御宿泊の予定でしょうか。
わたしと妻と子供一人の合わせて三人なのですが、三人部屋はあるでしょうか。
はい、三人部屋でしたらございますが、しかしお子様の年齢はおいくつですか。
五歳です。
そのようでしたら、ツインのお部屋にエキストラベッドをお入れすることもできますが。

図 15: 実験対象の例

試験対象	平均語長	再現率	適合率	交差数	文正解率
全体 (2352 文)	11.6 語	92.1%	92.9%	0.63	80.4%
一つの木構造として解析された文 (2049 文)	11.1 語	94.8%	94.7%	0.47	86.3%

表 1: 句構造解析の実験結果

再現率 (= 適合率)	文正解率
79.4%	48.8%

表 3: 格構造解析の実験結果

8 おわりに

本稿では、構文解析ツールキットを用いた句構造解析、依存構造解析、及び格構造解析の三つの解析実験の概要と実験結果を報告した。また、依存構造解析や格構造解析結果を定量的に評価するために、一般的な木構造を対象とする新しい評価尺度を提案した。

今後は、開発途中の格解析辞書や部分木併合規則を完成させるとともに、より効果的な探索手法や、より高度な言語知識を利用した実験を行なう。また、今回提案した木の評価法を用いて、文法規則を自動的に抽出する手法や、文法規則を自動的にチューニングする手法等も検討する予定である。

参考文献

[1] 田代, 森元: 音声言語処理のための構文解析ツールキット, 情報処理学会自然言語処理研究会資料, 106-12, 1995

[2] Jensen, K. and Heidorn, G.E: "The Fitted Parsing : 100% Parsing Capability in a Syntactic Grammar of English," ANLP83, 1983.

[3] 長尾 確, 丸山 宏: 自然言語処理における曖昧さとその解消, 情報処理, Vol.33, No. 7, 1992.

[4] Kita, K et al.: "Continuously Spoken Sentence Recognition by HMM-LR," ICSLP-92, pp.305-308, 1992.

[5] Morimoto, T. et al. : "A Speech and Language Database for Speech Translation Research", ICSLP94, 1994.

[6] Nagata, M.: A Stochastic Japanese Morphological Analyzer Using a Forward-DP Backward-A^* N-Best Search Algorithm, COLING-94, pp.201-207 (1994).

[7] Black, E., et al.: "A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars", DARPA Speech and Natural Language Workshop, 1991.

付録

A 解析木の評価手続き

```
type {木構造をノード構造とアーク構造で表現する}
        node = record {ノード構造}
                label               : ノードのラベル;
                arclist             : アーク構造のリスト;
        end
        arc = record {アーク構造}
                label               : アークのラベル;
                value               : ノード構造;
        end
procedure evaluate( std-tree , sys-tree : node )
var
        best,tmp: integer;
begin
        best := 0;
        foreach tree1 と tree2 を構成するすべてのノードの二つ組(NODE-i , NODE-j) に対して do begin
                tmp := compare-node( NODE-i , NODE-j ); { 最大スコアを返す組合せを求める }
                if tmp > best then best := tmp;
        end
        RECALL:= best /{正解木のノードの数};  PRECISION := best /{出力木のノードの数};
end
function compare-node( node1 , node2 : node )
var
        score, tmp, best                : スコアを表す数値;
        new-arclist, same-label-arcs    : アーク構造のリスト;
        best-arc                        : アーク構造;
begin
        if node1 のラベルと node2 のラベルが異なっていれば  then
                return 0;
        score := 1;
        if 二つのノードのいずれかが終端(葉)であれば then
                return  score;
        new-arclist := copy( node2.arclist );
        foreach arc1 in node1.arclist do begin
                best := 0;
                same-label-arcs := find-same-label-arcs( arc1.label , new-arclist );
                                {arc1 と同じアークラベルを持つ node2 の アークの集合を求める}
                foreach arc2 in same-label-arcs do begin
                        tmp := compare-node( arc1.value , arc2.value );
                        if tmp > best then begin
                                best = tmp; best-arc = arc2;
                        end
                end
                new-arclist = delete( best-arc , new-arclist );
                {局所的に最大のスコアを返した arc は探索対象から外す。}
                score = score + best;
        end
        return best + score;
end
```